
# A MODEL FOR COMPARING GAME THEORY AND ARTIFICIAL INTELLIGENCE DECISION MAKING PROCESSES

THESIS

Paul R. André
Captain, USAF

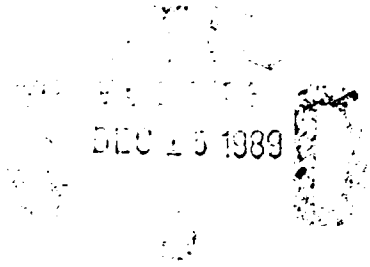
AFIT/GSO/ENS/89D-1





DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GSO/ENS/89D-1

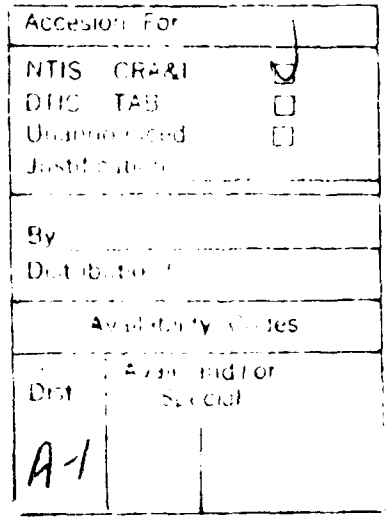



# A MODEL FOR COMPARING GAME THEORY AND ARTIFICIAL INTELLIGENCE DECISION MAKING PROCESSES

THESIS

Paul R. André
Captain, USAF

AFIT/GSO/ENS/89D-1




Approved for public release: distribution unlimited

AFIT/GSO/ENS/89D-1

# A MODEL FOR COMPARING GAME THEORY AND ARTIFICIAL INTELLIGENCE DECISION MAKING PROCESSES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Space Operations

Paul R. André, B.S.
Captain, USAF

December, 1989

## *Preface*

The purpose of this study was to develop an analytical tool to compare the use of an artificial intelligence decision making method to a game theory decision making method. The game theory method used was the two-player zero-sum gaming technique. The artificial intelligence method used was the minimax technique with an alpha-beta cutoff heuristic. The simulation program was written in the C programming language. The artificial intelligence method made calls to a forward chaining shell called C Language Production System, CLIPS.

The original game theory method was developed by Capt Robert Palmer. It was written in Turbo Pascal and required some modification before use in this study. The program was converted to the C language, modified to fit its intended use, and fit with the artificial intelligence method described above.

I had a great deal of help and support throughout this thesis. I extend my sincere thanks to my faculty advisor, Maj Bruce Morlan, for his advise and encouragement during this trying period. My tnanks also go out to my thesis reader, Lt Col Valusek, whose patience in reading and correcting this report was greatly appreciated. Last, but by no means least, I wish to express my thanks to my wife, Linda, and children, Duane and Anne Marie, for putting up with me. Their understanding and love helped pull me through periods when I was ready to throw in the towel.

Paul R. André

# *Table of Contents*

## *List of Figures*

## *List of Tables*

## *Abstract*

The purpose of this study was to develop an analytical tool to compare the use of an artificial intelligence decision making method to a game theory decision making method. The game theory method used was the two-player zero-sum gaming technique. The artificial intelligence method used was the minimax technique with an alpha-beta cutoff heuristic. The simulation program was written in the C programming language. The artificial intelligence method made calls to a forward chaining shell called C Language Production System, CLIPS.

The original game theory method was developed by Capt Robert Palmer. It was written in Turbo Pascal and required some modification before use in this study. The program was converted to the C language, modified to fit its intended use, and fit with the artificial intelligence method described above.

The program played reconnaissance satellites against ground mobile Strategic Relocatable Targets (SRTs). It implemented the methods described within this document to allow for a comparison of the two techniques. For an initial comparison, the measure of effectiveness was the number of times the satellite incorrectly predicted the SRT's movement.

# A MODEL FOR COMPARING GAME THEORY AND ARTIFICIAL INTELLIGENCE DECISION MAKING PROCESSES

## *I. Introduction*

### *1.1 Background*

The armed services continuously look for methods to improve their war-fighting capabilities. As the complexity of weapon systems increase, the ability to evaluate the benefits and employment techniques for such a system grows increasingly difficult. When a new system is developed, questions arise:

1. How would this system affect the outcome of a war fought today;
2. How effective is a certain employment strategy against enemy forces; and
3. How are personnel trained under current budget constraints.

Part of the answer to questions like these is found through the use of computer simulations.

Computer simulations allow the user to model specific situations. They produce results which, when analyzed, help answer questions not easily defined. Simulations also work well with situations either impossible or impractical to duplicate in the field (5:5-6). These situations include factors such as satellites and nuclear weapons.

Consider answering questions on nuclear warfare. What targets receive high priority? What targets receive low priority? These questions cannot be answered

by trial and error. An analytical solution is obtainable, but doesn't allow the user to test different options in a timely manner. Simulations, on the other hand, are tools designed to help answer complex questions in a timely manner without actually performing the operation.

Another example which relates to the purpose of this thesis is satellite operations. Satellites are expensive to build, launch, and maneuver. Maneuvering a satellite for training purposes is expensive and reduces the amount of time the satellite performs operational missions. Maneuvering or launching a new satellite to test an employment strategy is both costly and time consuming. A computer simulation, on the other hand, would allow the planner to test employment strategies before use on an existing system. It would also allow training to take place without risk to the actual system or loss of operational use time.

War simulations support a spectrum of tasks including analysis of given situations, insight into the use of various strategies, and procedural training. One example is the Arsenal Exchange Model (AEM). It provides an analysis of strategic nuclear war between two opposing strategic forces (2:285). On the other end of the spectrum lies the Combined Arms Tactical Training Simulator (CATTS). It provides a realistic combat environment to allow soldiers to gain experience in combat operations (2:226-227). Although both of these examples demonstrate a single use, many simulations address multiple purposes.

The PALANTIR, created by Capt Robert Palmer, had two uses: to train personnel on the use of satellites and to provide insight into the movements of strategic relocatable targets (SRTs) (11:24). It simulated reconnaissance satellites searching for SRTs such as rail-mobile SS-24 Inter-Continental Ballistic Missiles (ICBMs). The satellite attempted to find the SRT while the SRT attempted to avoid detection. The simulation used the two-player zero-sum game theory approach to reach decisions (11:23). This approach is discussed further in Chapter 2.

Although PALANTIR used game theory players on both sides, some recent simulations use the artificial intelligence technique of rule-based players. LTG William Thurman states, "AI systems are critical to almost every situation where large quantities of information are being managed, including battle management, [and] training." (17:33) This describes one of the problems of war simulations; they use large quantities of information to decide what move is best in a given situation. This thesis incorporates artificial intelligence into the PALANTIR war game.

## *1.2 Specific Objective*

The purpose of this research was to develop a program to demonstrate the use of an artificial intelligence rule-based player versus a game theory zero-sum player in a war simulation. This thesis modified PALANTIR to include the use of rule-based players. The automated players use either a game theory method or an artificial intelligence method to make a decision on where to move next. A solution method is selected for each player at the start of the simulation and used by that player throughout the entire run.

The program provided the basis for a comparison of two techniques: the two-player zero-sum technique and the artificial intelligence minimax technique. This comparison included an assessment of the difficulties of implementing the players, the advantages and disadvantages of using each technique, and possible mixtures of the two for a hybrid technique .

## *1.3 Limiting Assumptions*

This thesis used the SRT versus reconnaissance satellite search problem to compare game theory and artificial intelligence decision making techniques. To keep the model simple and unclassified, four major assumptions were made. The following paragraphs describe these assumptions. The validity of using these assumptions is discussed in Chapter 6.

*1.3.1 Peace Time Operations.* The conflict is assumed to take place during peacetime. This assumption limits the reconnaissance platforms to only satellites. Thus, the moves between SRT sites are assumed to occur between satellite overflight times. The SRT commanders have access to satellite orbit information made available by the United Nations Convention on Registration (1:15-3). With these assumptions, the actual SRT move along the path connecting the two locations need not be considered because it's assumed to occur when there are no satellites overflying the area.

*1.3.2 Two-Dimensional Euclidian Space.* The SRT operates within a network spread across a flat surface area of limited size. The size limitation is due to the SRT movement restrictions, command and control considerations, nuclear weapons control, and other related reasons (8:12). The flat area was used to disallow terrain masking (blocking a satellite's view by using terrain features such as canyons).

*1.3.3 Decision Cycle.* Each decision cycle, the satellite makes one path decision and the SRT makes one follow-on site decision. The assumption is that the decision cycle lasts the same amount of time as the time between overflights. This time is found by dividing the period of the orbit by the number of satellites in the fleet. Currently, the decision cycle is set to 3 hours between possible moves.

*1.3.4 Fixed Satellite Paths.* Each satellite possesses a set of possible paths over the SRT operating area. The paths are parallel and are assumed to remain in a fixed position relative to the SRT sites. They do not change location or inclination during the simulation.

## *1.4 Summary*

In this chapter, the general background, specific objective, and limiting assumptions of this thesis were stated. The thesis compares two methods for making a decision within a model: an artificial intelligence method and a game theory method. A general description of each method is given in the next two chapters, starting with the game theory approach.

# *II. Game Theory*

The purpose of this chapter is to review the techniques used in solving two-player zero-sum games. Rather than produce a solution to the entire game, the technique solves the problem one move at a time. Two-player zero-sum games form the basis of many game theory techniques.

## *2.1 Two-Player Zero-Sum Games*

Two-player zero-sum games contain only two opposing players. Each player tries to defeat the other through a series of moves called strategies. The winnings of one player and the losses of the other sum to zero. In other words, what one player wins, the other loses (9:12).

A player chooses from many possible pure strategies. Each pure strategy involves one or more legal actions. The offense possesses one set of pure strategies and the defense another set. Each combination of one offensive pure strategy and one defensive pure strategy results in a payoff to the players (6:368). Thus, the choice of a strategy depends upon the payoffs for the available strategies and the goal of the player. "The goal may be to maximize the winnings, or to minimize the losses." (9:13) In either case, the goal of a player remains constant throughout two-player zero-sum game; only the strategies change.

## *2.2 Payoffs*

In game theory, a payoff represents the value the players associate with the outcome of the two specific strategies. The players classify the outcome as a win, a loss, or a draw. While zero payoffs represent a draw, the value of a win or loss depends upon the worth of the outcome to that player. Determining this value is not a trivial task (6:368). As Major Rollin Lutz puts it:

> How much for example is a human life in combat worth ... one tank, a battalion of artillery, or maybe a single hill. Placing values on such things is exceedingly difficult. (9:14)

### *2.3 Game Matrix*

To keep track of the available strategies and their relationship to opposing strategies, game theory uses a matrix. The rows relate to the possible strategies for Player I (offense), and the columns relate to the possible strategies for Player II (defense). The number placed in each row/column intersection represents the value Player I places on the outcome of those two strategies (19:G-2). While Player I prefers a high outcome, Player II prefers a low outcome. Player II's preference comes from the knowledge that in a zero-sum game, what the offense loses, the defense wins. In the game matrix, the payoffs for Player II are the negative of the corresponding payoffs for Player I (11:10).

Figure 1 shows a game matrix with three strategies for Player I and four strategies for Player II. If Player I chooses strategy B and Player II chooses strategy F, the resulting value to Player I of that move is $P_{BF}$. Figure 2 shows Player II's corresponding matrix.

Player II

| Player I | D | E | F | G |
|---|---|---|---|---|
| A | $P_{AD}$ | $P_{AE}$ | $P_{AF}$ | $P_{AG}$ |
| B | $P_{BD}$ | $P_{BE}$ | $P_{BF}$ | $P_{BG}$ |
| C | $P_{CD}$ | $P_{CE}$ | $P_{CF}$ | $P_{CG}$ |

Figure 1. Player I's Matrix

Player I

| Player II | A | B | C |
|---|---|---|---|
| D | $-P_{AD}$ | $-P_{BD}$ | $-P_{CD}$ |
| E | $-P_{AE}$ | $-P_{BE}$ | $-P_{CE}$ |
| F | $-P_{AF}$ | $-P_{BF}$ | $-P_{CF}$ |
| G | $-P_{AG}$ | $-P_{BG}$ | $-P_{CG}$ |

Figure 2. Player II's Matrix

Once the game matrix is obtained, the player must solve it to decide what strategy to play. The solution method depends upon the nature of the matrix. It may consist of a pure strategy or a mixed strategy (6:372-375). Each of these is explained in the following pages. In either case, the first step in finding the solution is to remove any dominated strategies.

### *2.4 Dominated Strategies*

A dominated strategy is a strategy whose payoffs for each of the opponents strategies are equal to or less favorable than those of an alternate strategy. Since the dominated strategy scores equal to or worse than the alternate strategy in all cases, the player will never use it. Thus, this strategy is removed from the game matrix (6:371).

Figure 3 shows an example of a game matrix with dominated strategies. Player I's goal is to maximize the payoffs while Player II's goal is to minimize them. For Player I, strategy A dominates strategy B; in each column, the payoff of strategy A is larger than the payoff of strategy B. For Player II, strategy F dominates strategy E, and strategy G dominates strategy D; in each row, the payoff in the dominant strategy is smaller than the corresponding payoff in the dominated strategy. Figure 4 shows the game matrix with all dominated strategies removed.

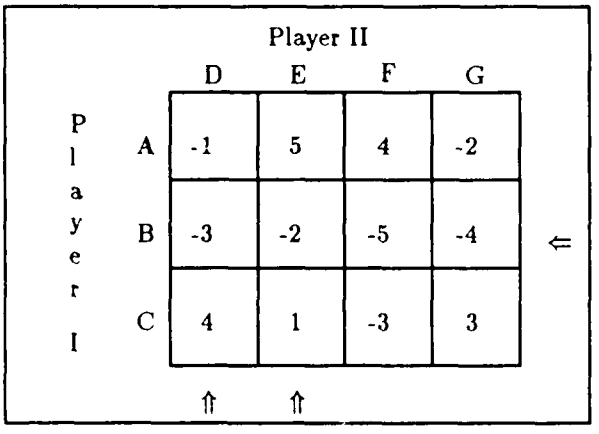



Figure 3. Typical Game Matrix

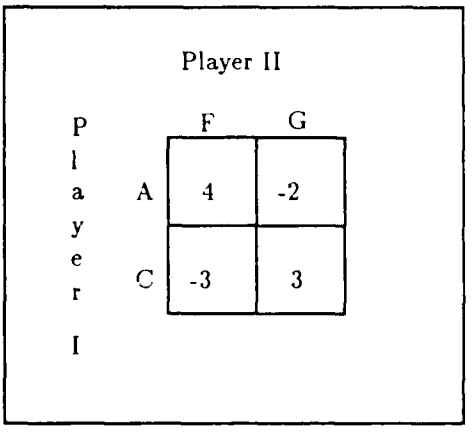



Figure 4. Reduced Matrix

Removal of dominated strategies from the game matrix makes it easier to solve. The first step of the solution technique involves looking for pure strategies. Saddlepoints within the matrix indicate a pure strategy exists.

### *2.5 Saddlepoints and Pure Strategies*

The simplest form of a pure strategy solution occurs when the matrix reduces to a single row and column after all dominated strategies are removed. In this case, each player's remaining strategy represents the only logical move to make under the given circumstances. Normally, a game matrix reduces to more than one row and one column.

To find a pure strategy in a complex matrix, the player first locates the saddlepoint of that matrix. A saddlepoint in a matrix occurs when each player's selection is based on the same element. To find this saddlepoint, game theory assumes each player chooses the best payoff from a list containing each strategy's worst payoff. Player I compares the minimum payoff of each row and selects the row with the maximum of these minimum payoffs. Player II, on the other hand, compares the maximum payoff of each column and selects the column with the minimum of these maximum payoffs. Together, these are called the minimax criteria. If these criteria result in the selection of the same matrix element, a saddlepoint exists (6:372-373).

Figure 5 shows an example of a game matrix with a saddlepoint. Using the above minimax criteria, player I selects row A and player II selects column F. Both players based their selection on the matrix element associated with row A and column F. This element is known as the saddlepoint of the matrix. The associated payoff for these strategies (-1) is called the value of the game.

The row and the column of the saddlepoint are the optimal or pure strategies for this move. Assuming the opponent uses the same logic, each player would only worsen the outcome by using another strategy. Thus, "neither player has any motive to consider changing strategies, either to take advantage of his opponent or to prevent

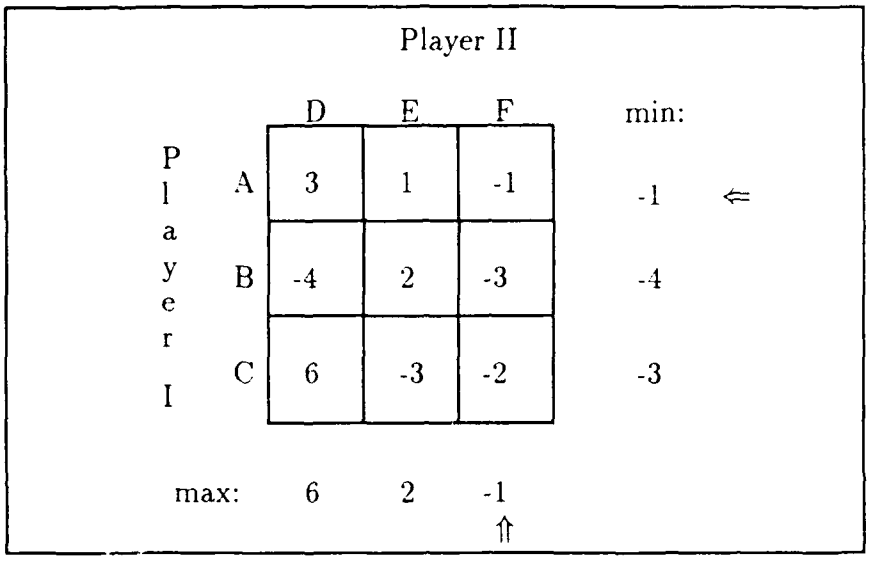



Figure 5. Game Matrix with Saddlepoint

the opponent from taking advantage of him." (6:373) In other words, the strategies represent a stable solution to this game matrix.

### *2.6 Mixed Strategies*

When a saddlepoint does not exist, the players select a strategy based on a set of probabilities. They assign a probability to each of the strategies and then randomly select one. The higher the probability, the greater the chance of selecting the strategy. These probabilities and their associated strategies are collectively called mixed strategies (14:509-510).

Linear Programming can be used to solve a game matrix for the optimal mixed strategy. The problem either minimizes or maximizes the value of the game depending upon whether it is solved for the offense (Player I) or the defense (Player II). When solving for the offense, the transpose of the game matrix is used. Thus, the value of the game is the cost function variable, each row is a constraint equation, and the set of probabilities form the constraint equations' variables (11:14).

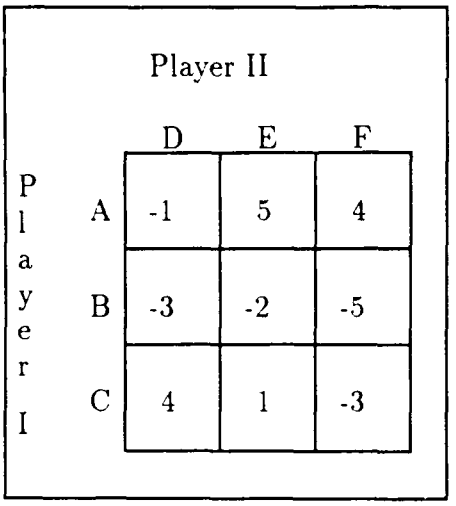



Figure 6. Game Matrix

Maximize: $X_4$

Subject to:

$$-X_1 - 3X_2 + 4X_3 - X_4 \geq 0$$

$$5X_1 - 2X_2 + 1X_3 - X_4 \geq 0$$

$$4X_1 - 5X_2 - 3X_3 - X_4 \geq 0$$

$$X_1 + X_2 + X_3 = 1$$

$$X_1, X_2, X_3 \geq 0$$

$X_4$ is Unrestricted

Figure 7. Linear Programming Problem

Three modifications to the matrix transform it into a linear programming problem: the addition of a cost function variable, the addition of a relationship to zero, and the addition of a constraint on the probabilities. The cost function variable is an unrestricted variable (can have a positive or negative value) subtracted from each row to link the cost function to the constraint equations. In a minimize problem, each constraint equation is set greater than or equal to zero. In a maximize problem, each constraint is set less than or equal to zero. The additional constraint equation requires the strategy probabilities sum to one (11:14). Figure 6 shows a game matrix while Figure 7 shows the resulting linear programming problem.

Once the steps are accomplished, the resulting linear programming problem is ready to solve using the simplex method. For the linear programming problem in Figure 7, the solution is $X_1 = .583$, $X_2 = 0$, $X_3 = .417$, and $X_4 = 1.08$. This results in a distribution for the game matrix in Figure 6 possessing a probability of using strategy A equal to .583 and a probability of using strategy C equal to 1 - .583. Strategy B is not desirable as it is dominated by strategy C. To select the strategy to use, a uniform random draw between 0 and 1.0 is performed. If the number is within a strategy's range (0-.583 for strategy A and .583-1.0 for strategy C), that strategy is selected.

## *2.7 Summary*

This chapter reviewed the game theory method of a two-player zero-sum game. It showed the basic setup of the game and explained the solution methods used for a single move. The first step in the solution method removes all dominated strategies from the game matrix. Next, the program searches for a saddlepoint. If found, the solution consists of two pure strategies, one for the offense and one for the defense. If no saddlepoint exists, the game matrix is converted into a linear programming problem and solved using the simplex method. This gives a probability distribution for the available strategies. A single strategy is selected from these mixed strategies based on the probability distribution. This strategy is then output as the solution to the game theory method. The artificial intelligence method is reviewed in the next chapter.

# *III. Artificial Intelligence Technique*

The purpose of this chapter is to review the artificial intelligence techniques used to solve games. Like game theory, the technique solves the problem one move at a time. Contrary to game theory, the artificial intelligence program is easily modified to fit changes in the overall concepts of operation (7:7).

## *3.1 Game Playing in Artificial Intelligence*

As with all other modeling techniques, artificial intelligence techniques determine the best move to make by examining the consequences of the possible moves. The difference between it and others is the process involved in getting the solution. A program using these techniques must determine the state of the game after a series of moves which start with the current move in question. It employs a move generation process to enumerate possible moves, an evaluation process to evaluate the resulting state of the game, and a selection or pruning process to eliminate unproductive moves (19:G-2,3). All of these processes occur simultaneously to determine the best move the player can make.

## *3.2 Game Tree*

The series of moves available to a player is maintained in a game tree The tree begins with a root node representing the current state of the game. Each branch from that node represents a possible move for a specific player. The node at the end of a branch represents the state of the game after that move takes place, and the branches out of this node represent the possible moves for the opponent. A ply of a tree refers to two consecutive levels of the tree. The first level represents the possible moves for one player while the second level represents the possible moves for the other player. Nodes on the last level of the tree are referred to as terminal nodes or leaves (12:319-321).

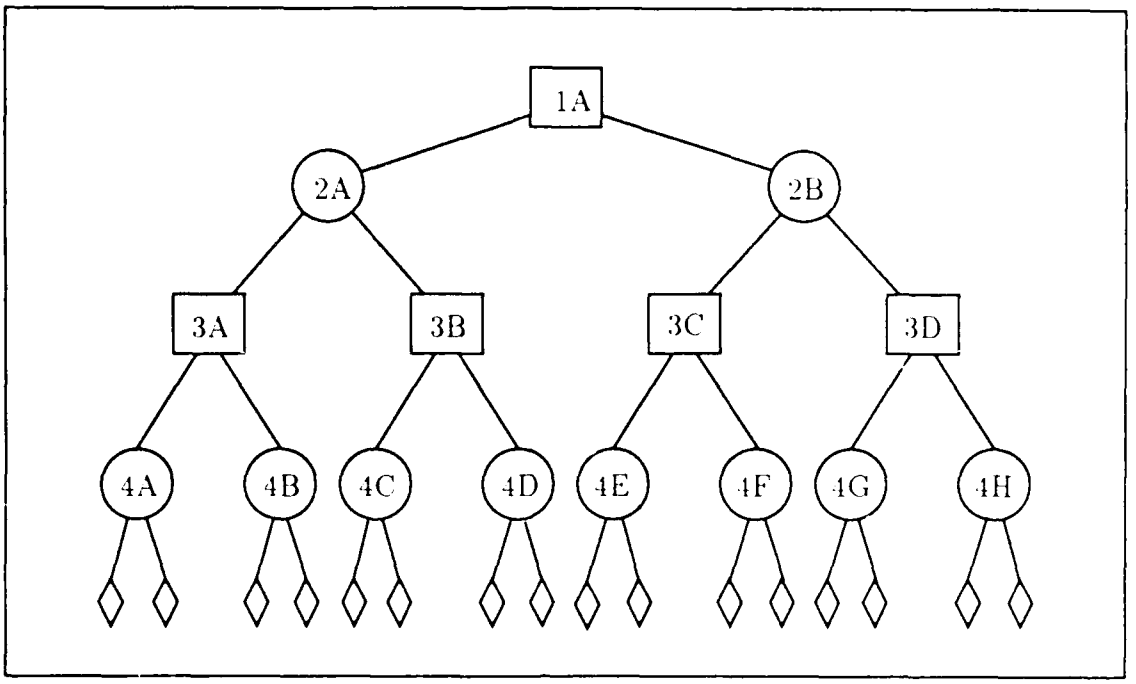



Figure 8. Two Ply Game Tree

Figure 8 shows a game tree with two ply. The square nodes represent decision points for Player I. The circular nodes represent decision points for Player II. The diamond nodes represent the terminal nodes of the tree. At each decision point the player has a choice of two moves. Although this tree is symmetric with two branches per node, game trees in general are asymmetric; the number of branches may vary depending upon the move generation process.

### *3.3 Move Generation*

The artificial intelligence method uses two enumeration techniques to generate moves: game-based and knowledge-based. "Game-based enumeration uses the rules of the game to generate options." (19:G-2) For example, a chess game program would produce all legal moves from the current board setup. In knowledge-based enumeration, move generation comes from knowledge of the goals, objectives, and tactics of each player. "Knowledge-based strategy is common for 'games' ... where

there are far too many legal moves to examine." (19:G-2) This method eliminates options not permitted by policies or directives (19:G-2).

Once a technique is selected, two basic methods exist for generating the actual moves: breadth-first and depth-first. Breadth-first generates all possible branches from the current level in the game tree before moving to the next level (3:256). Depth-first differs in that when given the choice of generating a move to a lower level or generating a move to the same level, it will generate the move to the lower level (3:251). For the tree in Figure 8, the breadth-first method generates 1A, 2A, 2B, 3A, 3B, 3C, 3D, and so on. Using the same tree, the depth-first method generates 1A, 2A, 3A, 4A, 4B, 3B, 4C, 4D, and so on.

In simple game trees with few ply, both breadth-first and depth-first work well, but in a complex tree with many ply, they are impractical. They require too much time to generate all possible moves. Two ways of avoiding this problem are limiting the number of ply generated and pruning the tree. Limiting the number of ply allows the method to generate enough branches to evaluate the current move while keeping the processing time down. Pruning the tree reduces the number of branches requiring enumeration (4:113). The first step in pruning the tree uses an evaluation process to assign a value to the branches.

### *3.4 Evaluation*

The evaluation process is very subjective. The programmer attempts to implement "the skills and tricks of clever play into the program, usually in heuristic form." (4:113) With a simple tree where all branches have been generated, the evaluation process is straightforward. It determines which bottom nodes lead to wins and selects the path which maximizes the winnings (4:113-114).

For a complex tree, the evaluation process uses a *static evaluation function*. At each bottom node, the function estimates the value of the state of the game for

that node. Both the decision process and the pruning process use this value in their computations (15:81).

The evaluation process can include an examination of any hidden options. Hidden options are moves not known to the opponent, but known to the player. The evaluation process determines the opponent's likely move without knowledge of the hidden option, and then determines the player's move with that knowledge. Players with knowledge of a hidden option can significantly increase their advantage over their opponent. The process must include the probability that the opponent knows of the option. The mathematics for including this probability was developed by R. Seldon in 1975 (19:G-4).

### *3.5 Pruning the Tree*

Both the evaluation process and the nature of conflicts provide information which can reduce (prune) the game tree to a manageable size (4:114-115). The pruning process can use a plausible move generator, a minimax procedure with a pruning heuristic, or both to significantly reduce the tree.

*3.5.1 Plausible Move Generator.* A plausible move generator disregards any possible move which is either unproductive or dangerous. Unproductive moves include those which provide no benefits to the player. Dangerous moves include those which put the game in jeopardy for no reason. Both produce no advantage over the opponent, and in some cases, produce a disadvantage. Generating only the plausible moves greatly reduces the game tree. If in a game of chess the plausible move generator reduced the moves per ply from 30 to 10, the branches for a 3 ply tree would drop from 729 million to 1 million (4:118). Because this process could cut moves where the player sacrifices some advantage to get a better advantage later, the programmer must use discretion during the move generator's implementation.

*3.5.2 Minimax Procedure.* The minimax procedure builds on the knowledge that "games involve two players determined to defeat each other." (4:115) It incorporates the static evaluation function described above and two heuristic rules. Morris W. Firebaugh describes these rules as:

- When A has the option to move, she will always choose the alternative which leads to the MAXIMUM benefits to her ... state with the largest $f_j$.
- When B has the option to move, he will always choose the alternative which leads to the MINIMUM benefit for player A ... smallest value of $f_j$. (4:115)

The minimax procedure assumes the opponent always makes a good move, only the values associated with a good move matter in the decision, and the heuristics remain constant throughout the game. Thus, a player with a choice of possible moves will select the move with the best predicted outcome (15:80-81).

In a game tree, a node with several branches is assigned the value of the branch with the best value. In this way, the values given to the bottom nodes by the static evaluation function propagate up the tree to the root node. During the propagation, heuristics such as the alpha-beta cutoff heuristic eliminate branches which have undesirable values (15:81).

*3.5.3 Alpha-Beta Cutoff Heuristic.* The minimax procedure uses the alpha-beta cutoff heuristic to prune the game tree. This heuristic is similar to the branch-and-bound technique used in Operations Research. During the evaluation of a tree, the partial evaluation of a branch sometimes indicates that the branch's best value is worse than the best value of a completely evaluated branch. In this case, the process will produce no useful information from finishing its evaluation of this branch. Therefore, it stops evaluating the branch and moves to the next one (4:118-119).

There are two pruning processes working at the same time: the alpha cutoff heuristic which trims moves of Player B and the beta cutoff heuristic which trims

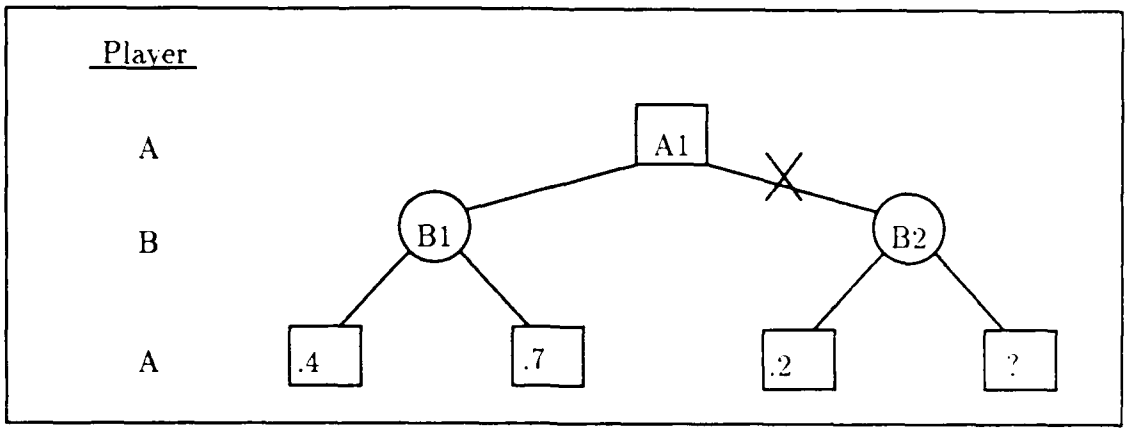



Figure 9. Sample Alpha Cutoff Heuristic

moves of Player A. The alpha cutoff is the maximum value evaluated at that level. The beta cutoff is the minimum value at that level (4:119). A good way to explain this process is with an example.

Figure 9 shows an example of the alpha cutoff heuristic. Player A is maximizing while Player B is minimizing. Using the static evaluation function, the branches of node B1 evaluated to .4 and .7. Since Player B is minimizing, the value given to node B1 is .4. This is called the alpha cutoff for that level. The first branch of node B2 evaluated to .2. Because Player B is minimizing, the value of node B2 must be equal to or less than .2. Since this value is less than the alpha cutoff, the process stops evaluating the branches of node B2 and prunes the B2 subtree from the game. If player A were given the chance to play node B2, player B would choose a move that gives at most a value of .2 to the node. Because node B1 provides a better value than node B2 does for player A, pruning the B2 subtree from the game before completely enumerating its branches does not affect the outcome.

Figure 10 shows a similar example for the beta cutoff heuristic. Again, player A is maximizing and player B is minimizing. Using the static evaluation function, the branches of node A1 evaluate to .4 and .2. Since Player A is maximizing, the value given to node A1 is .4. This is the beta cutoff for that level. The first branch of node A2 evaluated to .7. Because Player A is maximizing, the value of node A2

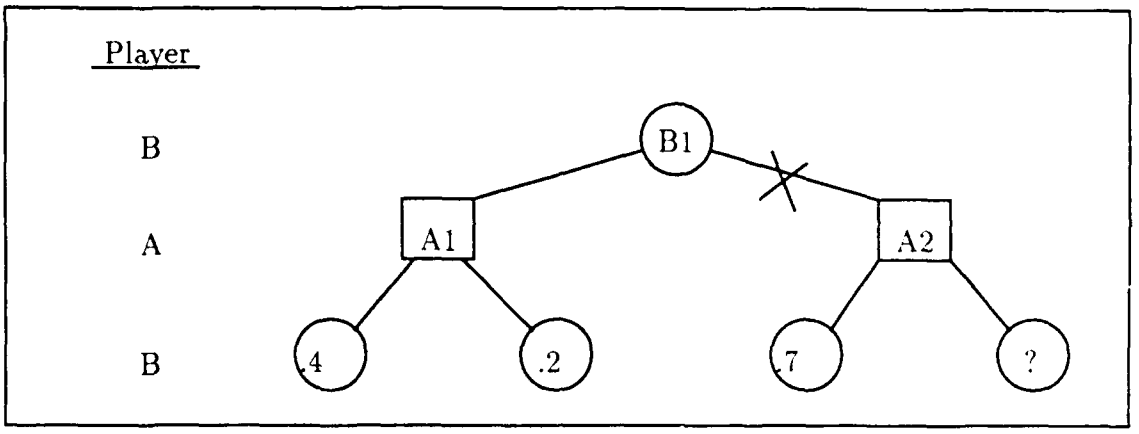



Figure 10. Sample Beta Cutoff Heuristic

must be greater than or equal to .7. Since this value is greater than the beta cutoff, the process stops evaluating the branches of node A2 and prunes the A2 subtree from the game. Even if the A2 subtree were left in the game, it would not affect the outcome as node A1 provides a better value.

### *3.6 Summary*

This chapter reviewed the artificial intelligence technique used in making decisions. The technique incorporates three basic processes: a move generation process, an evaluation process, and a tree pruning process. First, the generation process enumerates the possible courses of action from the current state of the world. Then the evaluation process places a value on the results of these moves. Finally, the tree pruning process eliminates those moves which are unproductive and possibly dangerous to the player. Based on the information these processes provide, the player selects the best move to make. The revisions to the original PALANTIR code necessary to implement the artificial solution method are discussed in the next chapter.

# *IV. Revisions to PALANTIR Program*

The original PALANTIR was designed as "an analytical tool to address strategic relocatable targets ... movement, to identify factors which influence SRT movement, and to predict possible SRT movements" (11:iv). It used the two-player zero-sum technique described in Chapter 2 for making all decisions. The purpose of this thesis was to develop a model to compare a game theory decision process, the two-player zero-sum technique, to an artificial intelligence decision process, the minimax technique described in Chapter 3. Since PALANTIR already had the framework for a model and the game theory process, it was chosen as the starting point for this thesis. Three modifications to the code were accomplished before the artificial intelligence method was added:

1. The program was rewritten in the C programming language;
2. The program was changed to a modular design; and
3. A simplex routine was added.

The first two modifications were made to ease the implementation of the artificial intelligence method. The third modification was made to implement a process discussed in Capt Palmer's thesis (due to time limitations, this process was not incorporated into the original PALANTIR). The addition of the artificial intelligence solution method, is described in Chapter 5. The three modifications are described in the following sections of this chapter.

## *4.1 Rewritten in the C Programming Language*

The code for PALANTIR was converted from Turbo PASCAL to Turbo C. This conversion was accomplished to allow a clean connection between the artificial intelligence routines and the rest of the program. The artificial intelligence routines are written in C Language Production System referred to as CLIPS.

The decision to use CLIPS for the artificial intelligence method was based on several reasons. First, it is a forward chaining shell. A forward chaining shell works from a list of facts to infer a conclusion based on a set of rules. In this thesis, the list of facts describe the current state of the satellite, the SRT, and the forecasted weather. The conclusion is either the path the satellite will follow or the site the SRT will move to. Second, CLIPS is readily available at the Air Force Institute of Technology. It was developed by NASA for use by government agencies. Finally, its routines are easily embedded within programs written in languages other than Lisp, especially the C programming language.

### *4.2 Modular Design*

For this thesis, PALANTIR requires more modularity than it possessed in the Pascal version. In a modular design, each routine acts as a black box; an input is entered into the routine and the desired output is produced. During the conversion to the C programming language, PALANTIR was rewritten to ensure all routines were of a modular design. A modular routine relys only on the variables input. It does not reference any variables defined in external routines unless the variable is passed to it. In the same way, the output of a routine is specified. The program does not modify any variable defined in an external routine unless the variable is passed to it.

The main area in PALANTIR requiring modification for modularity is the decision making process. Both the "SRT next site" decision and the "reconnaissance satellite path" decision are accomplished in the same module. PALANTIR used some of the information calculated for one player's decision to make the other decision. This interconnection causes problems when the SRT and satellite use different solution methods. One example of this was the use of the game matrix.

PALANTIR built one game matrix for use by both players. The matrix probabilities were based on the satellite elevation angle, the satellite characteristics, and

the forecasted cloud cover. Both the SRT and the satellite used this matrix to determine their moves. If the players use different solution methods, the shared game matrix approach no longer works. Thus, each player must possess the ability to build its own matrix. Another problem with using one matrix was the information used to calculate the matrix probabilities. Since the same matrix was used, both sides must possess the same information on the satellite characteristics, and both forecasted the weather exactly the same. This does not fit real world situations.

In the modified program, the modular design separates the solution procedure of the SRT from that of the satellite. Each player calculates the factors necessary to build a game matrix. From these calculations, each player builds and solves a game matrix based on the information they have available. Currently, the information available to each player is identical. In a future version this information could reflect the differences in available information and the forecasting techniques of the two sides. Separating the solution methods also improves the structure and readability of the program's code.

### *4.3 Simplex Routine*

The last modification to PALANTIR discussed in this chapter is the addition of a simplex routine. When no pure strategy exists, a distribution of the strategies available is produced and the actual strategy to use is randomly selected from this distribution. In PALANTIR, a uniform distribution of the strategies was used, and the strategy randomly selected from this distribution. This was done instead of the simplex routine described in the thesis documentation due to lack of time. For the modified program, a simplex routine was added to the game theory solution. The simplex routine solves a linear programming problem built from the game matrix. This process is discussed in Chapter 2. The resulting solution to the linear programming problem is the distribution of the strategies. As stated earlier, the strategy to use is randomly selected from this distribution.

## 4.4 Summary

The modifications described in this chapter were made to the code to enable the addition of the artificial intelligence routines. The modified code is an extension of the original PALANTIR. For a complete description of the implementation of the game theory method, refer to Capt Robert Palmer's thesis, *A Methodology Employing Competitive Strategies For Predicting Possible Relocatable Target Movements* (11:18 - 50). The last modification, the addition of an artificial intelligence solution method, is described in the next chapter.

# V. *Artificial Intelligence Implementation*

This chapter discusses the implementation of the artificial intelligence method. The rules are written for use by the C Language Production System, CLIPS. CLIPS is a forward chaining process as described below. It uses rules to build and evaluate the game tree. The alpha-beta cutoff heuristics are incorporated within the rules to reduce the size of the tree. The tree is further reduced by limiting the number of ply (the number of future moves looked at to determine the current move). This limit is input by the user during the initialization of a run.

## *5.1 The Forward Chaining Process*

A forward chaining process works from a list of known facts to infer a conclusion. First, all facts known about a set of circumstances are placed in a list. Next, the forward chaining process uses a set of rules to draw inferences from this fact list. The rules follow an "if...then..." structure; if a set of facts exist, then perform a series of actions. These actions include removing facts from the list, adding new facts to the list, and printing out intermediate results. When no more rules apply to the list of facts, the process ends. The process either outputs the results via a rule, or places the results in the fact list as a fact.

## *5.2 The Fact List*

The list of known facts comes from four sources: the setup information, the network information, the satellite information, and the weather information. Each source adds a set of facts to the fact list. The facts obtained from each source are explained in the following paragraphs.

*5.2.1 The Setup Information.* This information produces facts about the current status of the run. This includes the current time, the current SRT operating

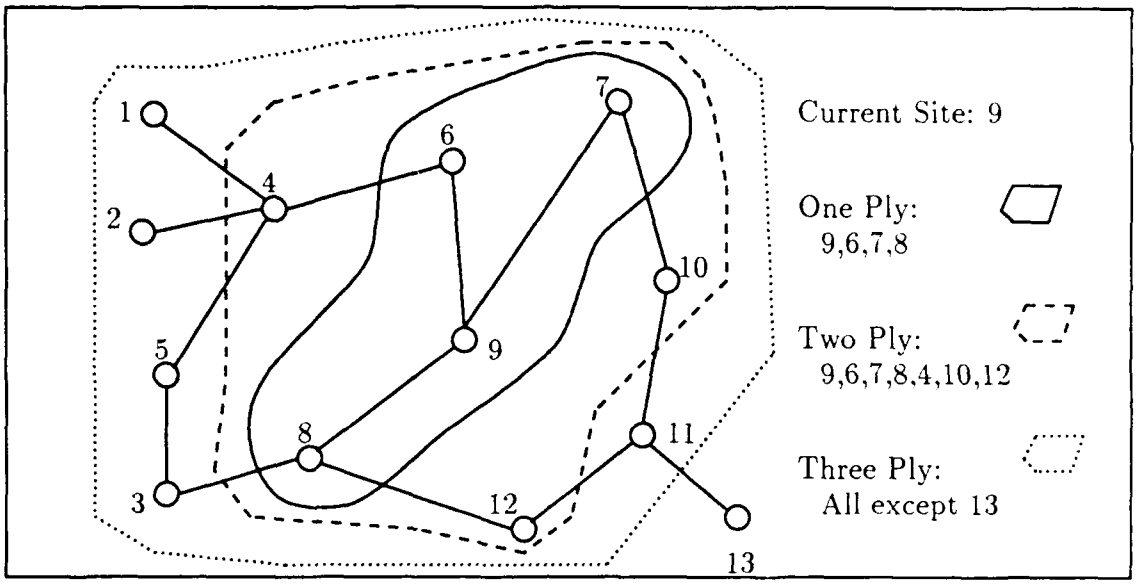



Figure 11. Sample Network

site, and the number of moves the process should look ahead to decide upon the current move. The last fact describes the depth of the tree. If the process is to look three moves into the future, the depth of the tree is three ply.

*5.2.2 The Network Information.* This information produces facts about a portion of the SRT network. The number of sites included in the fact list depends upon the depth of the tree. The first set of facts added to the list are those pertaining to the current SRT operating site. The next set of facts added pertain to each of the current branch sites (a branch site is a site directly connected to the current SRT operating site). These are the only facts required for a one ply tree. For each additional ply, another level of sites is added. Figure 11 shows a sample network and lists the sites whose facts are placed in the fact list for a one, two, and three ply tree. The facts for each site include the kilometers North of 40N, kilometers East of 160E, and a list of the site's branch sites.

*5.2.3 The Satellite Information.* This information produces facts about each satellite as well as facts about the satellite fleet. The facts for each satellite include

its path's slope, y-intercept, and time of overflight. The y-intercept is measured in kilometers from 160E to where the satellite's path crosses 40N going North. The facts for the satellite fleet include altitude, number of possible paths the satellite can take over the area, separation between the paths in kilometers, satellite efficiency, and minimum probability of detection of a target. The satellite efficiency incorporates both the efficiency of the reconnaissance hardware and the efficiency of the photo interpretation facilities.

*5.2.4 The Weather Information.* This information produces facts about the cloud cover over each of the sites in the fact list. A fact contains ten forecasts of the cloud cover over the site. Each forecast is for a time 3 hours after the last forecast. In this way, the cloud cover over a site is forecast for a 30 hour period.

### *5.3 The Satellite Game Tree*

The satellite game tree consists of one root node and several ply. The root node serves as a base for the first ply. Each ply has two levels: the B level and the A level. The B level contains nodes for the possible satellite paths; the A level contains nodes for the possible SRT follow-on sites.

A satellite tree starts with the root node. If the SRT was detected last turn, the root node represents the site where the SRT is located. If it was not detected, the root node represents the site where the SRT is thought to be located. The hope is the SRT will be redetected within the next few moves. The root node has a branch to each of the first ply's B level nodes. These branches represent possible paths the satellite can fly over the operating area. Each B level node then branches to a set of A level nodes. The sets are identical and contain one A level node for each possible SRT follow-on site. This completes the first ply. From this point on, each A level node acts as a root to a subtree following the same format as above. The A level nodes branch to sets of the next ply's B level nodes. Each set contains one B level

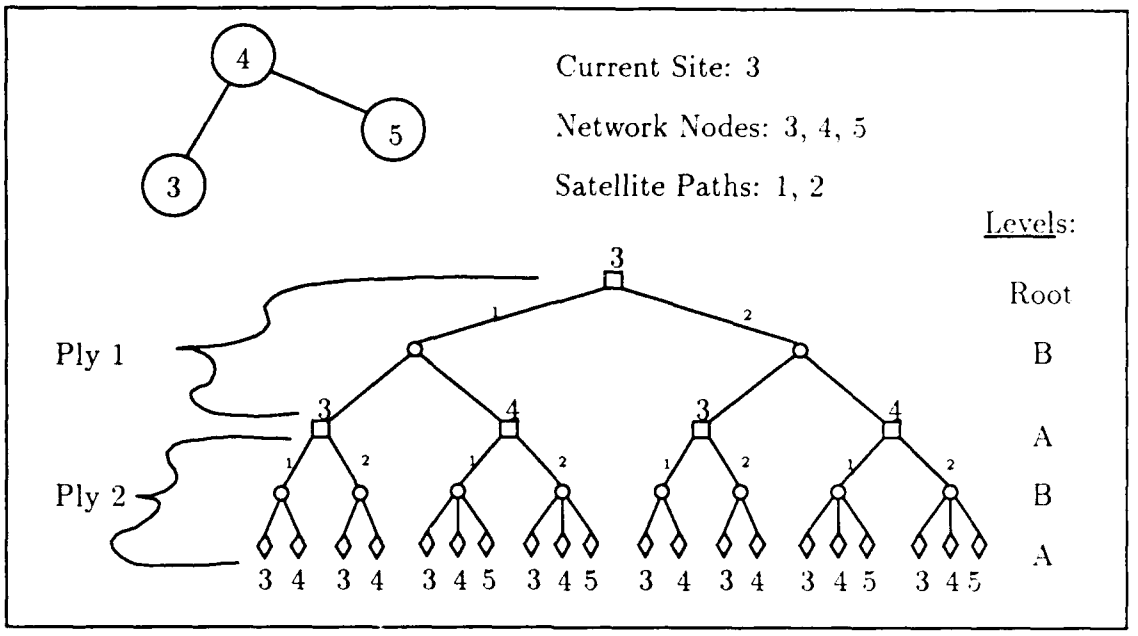



Figure 12. Satellite Game Tree

node for each possible path the satellite can fly. Again each B level node branches to a set of A level nodes. The set contains one A node for each SRT follow-on site of the subtree's root node. Within the subtree, these sets are identical. This process continues for the depth of the tree. Figure 12 shows a simple network of three nodes and the corresponding satellite tree.

### *5.4 The Strategic Relocatable Target Game Tree*

The SRT game tree is similar to the satellite game tree. The differences occur with the root node and the order of the levels. The root node represents the actual SRT operating site. The branches from the root node connect it to the first ply's A level nodes which represent the possible follow-on sites. The final difference is the order of the levels within the ply. The A level appears above the B level within a ply of the SRT game tree. Figure 13 shows the SRT game tree which corresponds to the satellite game tree and network in Figure 12.

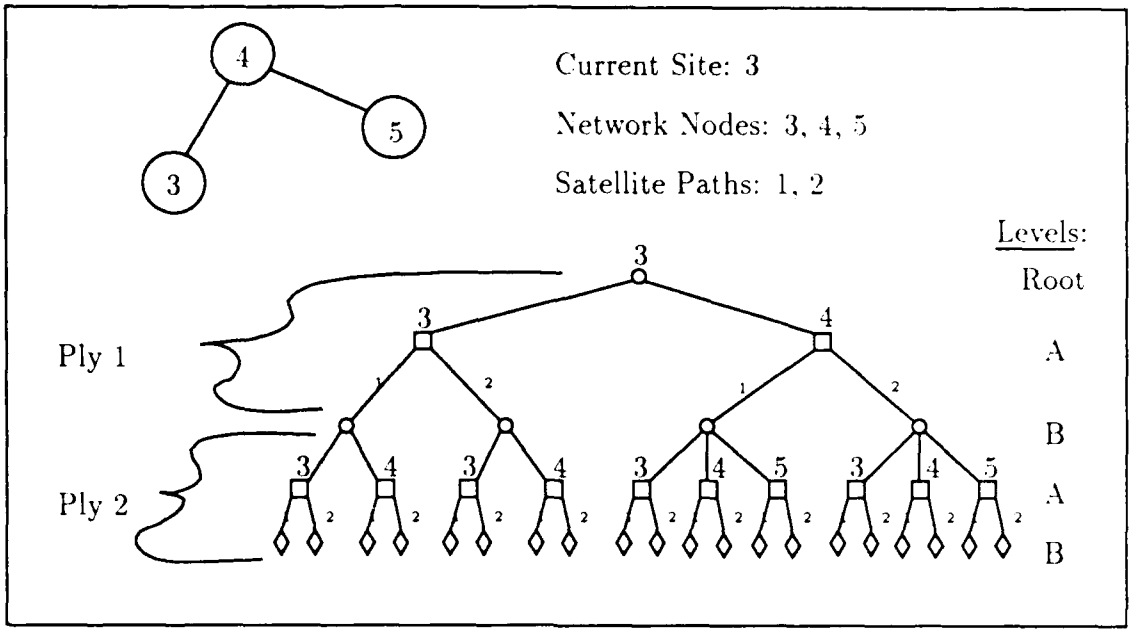



Figure 13. SRT Game Tree

### *5.5 The Evaluation Process*

The evaluation process assigns a value to a pair of connected A/B level nodes existing within the same ply. The pair represents one possible move in the simulation. The value of the pair depends on the site's coordinates, the path the satellite flies, and the forecasted cloud cover over the site. First, the code computes the angle of elevation between the site and the satellite. It then determines what the forecasted cloud cover is over the site at the desired time of detection. The elevation and forecast are used in a regression formula to compute the Cloud Free Line Of Site (CFLOS) value. This value is multiplied by the satellite efficiency and the percentage of illumination to obtain the probability of detecting an SRT at a particular site. The illumination percentage is 1.0 at noon and reduces linearly to a minimum illumination value at Midnight. The angle of elevation is measured from the horizon to the line between the site and the satellite at the point of its closest pass. This computation is simplified to an algebraic one involving no orbital mechanics.

The CFLOS computation was modified during this thesis to attempt a better fit. A description of the method and data used to obtain the equation is outlined in appendix A of this thesis. The regression equation is:

$$CFLOS(\theta, \mu) = .88964 + .0021217 * \theta - .75599 * \mu + .77157 * \mu^2 - .93593 * \mu^3 \quad (1)$$

where $\theta$ stands for the angle of elevation in degrees, and the $\mu$ stands for the forecasted percentage of cloud cover over the site. The percentage of cloud cover ranges from 0 for clear skies to 1.0 for completely cloudy skies.

### *5.6 Building the Game Trees*

Both trees are built in a depth-first manner. The left side of a tree is built first to the maximum depth of the tree. As the process continues, more nodes are added working from top to bottom and left to right. The alpha-beta cutoff heuristic is incorporated during the building process. This greatly reduces the size of the tree.

When the building process comes to a leaf (bottom-most nodes on a tree), it computes the probability of detection. It then propagates this value up the tree using the minimax technique described in Chapter 3. If a subtree is determined to be unproductive via the alpha-beta cutoff heuristic, it is trimmed from the tree, and the enumeration of its nodes is stopped.

### *5.7 Propagation of a Value*

Propagation of a value of a node occurs either after the node is completely enumerated or immediately if the node is a leaf. A node is completely enumerated when all of its branches are completely enumerated. To keep track of when to propagate a node's value, a counter is used. As each branch of the node completes

its enumeration, the counter is incremented by one. When the counter indicates all the branches are enumerated, the node's enumeration is complete. At this point, the process compares the node's value against the value of its parent. If the node's value is "better", the parent's value is set equal to it. If the node's value is not "better", the propagation does not occur. For a minimizing parent (B level node), the term "better" means the node's value is less than the parent's value. For a maximizing parent (A level node), the term "better" means the node's value is greater than the parent's value.

### *5.8 Pruning the Tree*

The pruning of a tree by the alpha-beta cutoff heuristics involves two processes: updating of the alpha-beta cutoff values and testing a node for pruning. In both cases, the value of the node determines the outcome. The first step is to update the cutoff values.

*5.8.1 Updating the Cutoff Values.* Each ply has an alpha cutoff value for the B level and a beta cutoff value for the A level. An A level node's value is set to the maximum value of its branches with the associated beta cutoff value being set to the minimum of these maximums. A B level node's value is set to the minimum value of its branches with the associated alpha cutoff value being set to the maximum of these minimums. The cutoff value for a particular level is initialized at the start of the enumeration of a set of nodes with a common parent. An alpha cutoff is initialized to zero while a beta cutoff is initialized to one.

As each node in the set completes its enumeration, its value is checked against the cutoff value for that level. If the value of a B level node is greater than the associated alpha cutoff value, it becomes the new cutoff value for that set. Likewise, if the value of an A level node is less than the associated beta cutoff, it becomes the new cutoff value for that set. Due to the initial values of the cutoffs, the first node

of the set will always replace the cutoff value. Once the enumeration process moves to the next set of nodes with a common parent (different from the first set's parent), the cutoff values are reinitialized.

*5.8.2 Testing and pruning.* The testing and pruning of a node is linked to the propagation of the values of its branch nodes. Every time the node receives a new value from one of its branches, it is tested against the cutoff value for pruning purposes only (the node's value can only replace the cutoff value after all of its branches are enumerated).

Each A level node is tested against the beta cutoff value for the ply the node lies in. If the value of the A level node is greater than the associated beta cutoff value, its remaining non-enumerated branches will never lower its value below the cutoff. This is due to the nature of the A level node; it takes the maximum value of its branches. Since the parent level node (a B level node) takes the minimum value of its branches, and since the value of this A node is already greater than a previously enumerated node (indicated by the lower beta value), this node has no effect on the decision process. The enumeration of its branches is stopped, and it is pruned from the tree. A similar process occurs for a B level node.

Each B level node is tested against the alpha cutoff value for the ply the node lies in. If the value of the B level node is less than the associated alpha cutoff value, its remaining non-enumerated branches will never raise its value above the cutoff. This is due to the nature of the B level node; it takes the minimum value of its branches. Since the parent level node (an A level node) takes the maximum value of its branches, and since the value of this B node is already lower than a previously enumerated node (indicated by the higher alpha value), this node has no effect on the decision process. The enumeration of its branches is stopped, and it is pruned from the tree. Examples of these cutoffs were discussed in Chapter 3.

## 5.9 *Solving the Game Tree*

The nature of this simulation does not allow the direct use of the minimax technique. The technique requires a modification for it to produce useful information. The problem stems from different branches possessing the same path/site combinations. Because the paths of a satellite remain constant relative to the sites, a path/site combination evaluates to the same probability of detection no matter its location within the tree. Therefore, the probabilities of detection computed at the leaves are independent of the moves taken to get to the leaves.

The modification to the minimax technique has two parts. First, the probability of detection is computed for each set of moves along the routes enumerated in the tree. A move set includes one satellite path decision and one SRT site decision. The decisions are represented by one A level node and one B level node connected in the tree within the same ply. Thus, for every path/site combination enumerated in a tree, a probability of detection is calculated. With the original technique, only the probability of detection for the leaves is calculated. The second part of the modification applies to the propagation of the probability up the tree to the root node. When each node is tested for the alpha-beta cutoff, the value used is the product of the probability of detections along the route from the node being evaluated to a leaf. This allows the entire route over the depth of the tree to influence the decision process. A route probability takes the form:

$$P_{Route} = P_{Node1} * P_{Node2} * P_{Node3} * \ldots * P_{Leaf} \qquad (2)$$

where

$P_{Route}$ = Probability of detection along the route,

$P_{NodeX}$ = Probability of detection at node X, and

$P_{Leaf}$ = Probability of detection at the last node of the route.

The solution to the move decision is the branch from the root node with the "best" value. For a satellite path decision, this branch is the one with the maximum product of the probabilities. For the SRT site decision, this branch is the one with the minimum product of the probabilities. Both selection processes account for the fact that nodes with an SRT site decision selects the branch with the minimum detection value while the nodes with a satellite path decision selects the branch with the maximum detection value.

### *5.10 Summary*

This chapter discussed the implementation of the artificial intelligence solution method in the revised PALANTIR program. The process uses the CLIPS forward chaining shell and a list of facts about the SRT network, satellite fleet, weather forecast, and general game information. A set of rules implement the minimax technique with an alpha-beta cutoff heuristic. This minimax technique was modified to allow the route within the tree to influence the selection process. Finally, the selection of a move was based on the results of this minimax technique. The next chapter discusses the verification and validation of this simulation model.

# *VI. Verification and Validation*

Before a model is used for its intended purpose, the user needs assurances that it works properly. Two processes are used to check a model's performance: verification and validation. The verification process ensures the model's code works as the programmer intended. The validation process ensures the model reacts the same as the real world system it was designed to simulate. Together, these two processes give the users the assurances required. This chapter discusses the verification and validation processes used for the PALANTIR model.

## *6.1 Verification*

As stated before, the verification process ensures the model's code works as the programmer intended. How the process is implemented varies from programmer to programmer. In this thesis, the process consisted of three steps:

1. Ensure each module of the program produces the expected output for a particular input;
2. Develop a flowchart from the code showing the interconnections between the modules; and
3. Define all of the variables used by the code and give a general description of the purpose of each major section of the code.

Each step was performed for the revised PALANTIR. During the process, numerous errors were found and corrected. After each correction, the module which contained the correction was reverified. Each step of this verification process is described in the following paragraphs.

*6.1.1 Input/Output Comparison.* For each module, the output produced by the code for a known input was compared to the expected output. First, the input to a module was printed before the module manipulated it in any way. After the module performed all of its manipulation of the data, the output was printed. Next, the input was "walked through" the code of the module by hand. Finally, the results of this "walk through" were compared to the actual output of the module and to the expected output of the module. If the results differed, the problem was tracked down and corrected, and the comparison was reaccomplished.

The order in which the modules were checked assured that only the current module's code was required. First, the modules which did not call any other modules within their code were run through the input/output comparison. After all of these modules were checked and corrected, the modules which only called modules previously checked were run through the input/output comparison. Using this process ensured that only the code of the module currently being checked was required. Once the input/output comparison was completed for all of the modules, the second step in the verification process was performed.

*6.1.2 Developing the Flowcharts.* A set of flowcharts was developed showing the interconnections between the modules. Once completed, the logic flow was checked and, if in error, corrected. The flowcharts were compared directly to the code. Appendix B contains four of the top level flowcharts produced during this step. The first flowchart shows the overall structure of the program. The remaining three flowcharts show the three major areas of the code.

The logic flow begins with an opening sequence. This contains an introduction, a listing of the major system variables, and the reading of the system inputs from disk files. After the opening sequence, the logic flow moves through a loop. The loop consists of planning the satellite move, planning the SRT move, and executing

both moves at the same time. The execution of the moves both updates the system variables and displays the results to the user.

*6.1.3 Defining Variables and Commenting Code.* The last step listed in the verification process actually began with the start of the writing of the code. As the code was written, each variable was explicitly defined and commented as to its use. Also, all of the major sections of the code were fully commented both to describe their general purpose and to explain the technique used. This ensured the code and variables were easily understood during the other two verification steps. It also ensured future understanding for later modification.

### *6.2 Validation*

As mentioned above, the validation process ensures the model reacts in a manner similar to the real world system it simulates. The ideal model simulates all aspects of the real world system. In most systems, the number of factors involved makes the ideal simulation impossible to achieve. Therefore, simplifying assumptions are made to the nonessential elements of the system while still allowing the detail required for the essential elements. The validation process ensures the model with its assumptions reacts as the real world system does within the areas essential to the study. The validation process used for PALANTIR involved three steps:

1. Build the model with a high face validity;
2. Validate the model's assumptions; and
3. Compare the model's input-output transformation to that of the real world system.

*6.2.1 High Face Validity.* The face validity of a model shows how well it responds to changes in the input. In other words, does the model react properly to changes in the input? Three tests were performed. The revised PALANTIR reacted

properly to changes in the input. First, the minimum illumination was raised and lowered. The probabilities of detection raised and lowered in response to this change as expected. Second, the satellite efficiency was raised and lowered. Again, the probabilities of detection raised and lowered in response to this change as expected. Finally, the cloud cover was changed to reflect the same cloud cover over the entire operating area. The probability of detection was compared to the distance from the path. As expected, the farther from the path the site was located, the lower the probability of detection. See appendix A for a complete discussion of the regression equation used for the CFLOS calculation.

*6.2.2 Assumption Validation.* The second step in the validation process is the validation of the major assumptions. As was indicated earlier, no model is built to fully simulate the real world system. Therefore, assumptions are made to simplify the problem and the code. A validated model requires that all assumptions made are valid. The following paragraphs state the validity of the major assumption listed in Chapter 1.

*6.2.2.1 Peace Time Operations.* The conflict is assumed to take place during peacetime. This assumption limits the reconnaissance platforms to only satellites. During peacetime, the Soviets will not allow US reconnaissance aircraft over their country, and the US does not fly missions of this type for political reasons. During wartime, the Soviets would still not allow such flights, but the US would be more willing to accept the risks. Therefore, this simulation takes place during peacetime, and the only reconnaissance platforms available to look at the operating area are satellites. Because of this, catching an SRT moving between two sites is highly unlikely. Satellite overflight times are assumed known to the Soviets because of the satellite information made available by the United Nations Convention on Registration (1:15-3). Thus, the actual move along the path connecting the two locations need not be considered.

*6.2.2.2 Two-Dimensional Euclidian Space.* The SRT operates within a network spread across a flat surface of limited size. The size limitation is due to the SRT movement restrictions, command and control considerations, nuclear weapons control, and other related reasons (8:12). The two-dimensional euclidian space was used to disallow terrain masking (blocking a satellite's view by using terrain features such as canyons). Terrain masking would greatly add to the complexity of the model and, since it is disallowed for both solution methods, the simplification should not affect the comparison.

*6.2.2.3 Decision Cycle.* Each decision cycle, the satellite makes one path decision and the SRT makes one follow-on site decision. The decision cycle lasts a specific amount of time. This is derived from the time between overflights of the reconnaissance satellites. When multiple reconnaissance satellites are used, the normal procedure is to place the satellites an equal distance apart to maintain an even surveillance. Thus, the time between overflights is a constant value. The value is equal to the period of the orbit divided by the number of satellites. Since the time is constant, the overflights occur in a cycle. Therefore, the use of a decision cycle that matches the time between successive overflights is valid. When classified information is available for use in a later version of PALANTIR, this assumption may change to match the new data.

*6.2.2.4 Fixed Satellite Paths.* Each satellite possesses a set of possible paths over the SRT operating area. The paths are assumed to remain fixed in relation to the SRT sites. The validity of this assumption lies in the United Nations Convention on Registration. Although minor orbital variations would not require a change in the orbital registration, a major change in the orbit would. Allowing the path parameters to reflect a centering on the previous path could after several iterations lead to a major change in the orbit. If the East most path was selected several times in a row, the satellite paths would change by several hundred kilometers to the

East. The assumption is the equivalent of allowing a small amount of fluctuation in the orbit without changing the overall orbit parameters. This fits with real world limitations.

*6.2.3 Input-Output Transformation Comparison.* The last step in validating the model is to compare the input-output (I/O) transformations of the model to that of the real world system. Similar inputs into the real world system should result in similar outputs. This is the most difficult step in the validation of PALANTIR due to the classified nature of the actual problem. To perform this validation on the revised code, the original PALANTIR was used. The original code had already passed the validation process and therefore provided an acceptable reference for comparison. The revised I/O transformation closely resembled that of the original code. At the present time, this will serve as a loose confirmation of the validity of the revised code for this step. A more accurate validation of the revised PALANTIR should be accomplished in the future using classified information.

## *6.3 Summary*

In this chapter, the revised code of the PALANTIR model was verified and validated for the intended purpose of this thesis. The verification ensures the model's code works as it was intended. The validation ensures the model behaves in a manner similar to the real world system. The validation process was accomplished with a full understanding of the purpose of this thesis. In the next chapter, the revised PALANTIR will be used for a first-cut comparison of the artificial intelligence and game theory decision making techniques described earlier in this thesis report.

# *VII. The Comparison*

This chapter describes the procedures used to compare the performance of the game theory technique to that of the artificial intelligence technique. This is an initial comparison of the two techniques as the main emphasis of this thesis was the development of the model. In PALANTIR, a game theory strategy consists of only one move. Therefore, for this comparison, the artificial intelligence technique was run using a game tree with only one ply. The comparison uses a single measure of effectiveness during the evaluation of four types of simulation runs. The measure of effectiveness, types of runs, and results are presented in the following paragraphs.

## *7.1 Measure Of Effectiveness*

The measure of effectiveness is the number of times the satellite does *not* accurately predict the SRT's follow-on site. The path the satellite will fly over the SRT operating area depends on this prediction. The probability of detecting the SRT is highest at the sites directly under the path flown and decreases with increasing distance from that path. Therefore, the satellite optimizes its path selection based on its prediction of the SRT's follow-on site. If the prediction is wrong, the probability of detecting the SRT can decrease. This decrease in probability of detection depends on the SRT's actual location relative to the path flown.

In a similar fashion, the SRT optimizes its follow-on site selection based on a prediction of the satellite's next flight path. If the SRT can predict which path the satellite will fly, it can move to the site which has the least probability of detection for that satellite path. In other words, it attempts to move to a site different from the site the satellite predicted to be the follow-on location. The performance of both decision processes is directly related to the number of times the satellite wrongly predicts the SRT's follow-on site.

### *7.2 Run Types*

Four types of simulation runs were used for the comparison. The run types differed based on the decision technique used by each player. Table 1 shows the different types of runs used for this comparison.

Table 1. Simulation Run Types.

| | | Satellite | |
|---|---|---|---|
| | | Game Theory | Artificial Intelligence |
| *SRT* | Game Theory | (GT,GT) | (GT,AI) |
| | Artificial Intelligence | (AI,GT) | (AI,AI) |

The first run type established a baseline used in comparing the artificial intelligence techniques in run types (AI,GT) and (GT,AI) with their game theory counterparts. Run type (AI,AI) established a baseline for comparing the game theory techniques in run types (AI,GT) and (GT,AI) with their artificial intelligence counterparts. Together, these two comparisons provided an overall comparison of the two decision making techniques.

### *7.3 The Results of the Simulation Runs*

During the analysis, the two decision techniques used the same evaluation equation. This allowed the analysis to compare the actual techniques rather than the evaluation functions. Fifty runs of each run type were performed with thirty decision cycles per run. The remainder of this section describes the results of the comparison.

Table 2 shows an analysis of the run types. When both the satellite and the SRT used the game theory technique (run type (GT,GT)), the number of incorrect site predictions averaged 1.76 per run with a variance of 0.879. When both the satellite and the SRT used the artificial intelligence technique (run type (AI,AI)), the number of incorrect predictions averaged 2.06 per run with a variance of 0.944. When the SRT used the artificial intelligence technique and the satellite used the

game theory technique (run type (AI,GT)), the number of incorrect site predictions averaged 2.08 per run with a variance of 1.040. When the SRT used the game theory technique and the satellite used the artificial intelligence technique (run type (GT,AI)), the number of incorrect site predictions averaged 1.14 per run with a variance of 0.758. This run type produced a significantly lower error rate than the other three. The lower number of errors and reduced variation indicated the satellite had an advantage over the SRT when using the artificial intelligence decision technique. At the significance level used, there was no difference between the other three run types; they could have come from the same population. The tests were conducted with a significance level of .01 (10:263-265).

Table 2. The Simulation Results.

| Run Type | Low | High | Mean | Variance | Distribution |
|---|---|---|---|---|---|
| (GT,GT) | 0 | 9 | 1.76 | 0.879 | Proportion |
| (AI,GT) | 0 | 10 | 2.08 | 1.040 | Proportion |
| (GT AI) | 0 | 7 | 1.14 | 0.758 | Proportion |
| (AI,AI) | 0 | 12 | 2.06 | 0.944 | Proportion |

### *7.4 Analysis*

All errors in the predictions occurred when two or more alternatives appeared as the best selection for different reasons. The artificial intelligence technique processed these situations in the same way it did when only one alternative appeared as the best selection. The game theory technique, on the other hand, treated the situation entirely different. It suggested a mixed strategy solution. A mixed strategy solution randomly selects an alternative based on a distribution of the best alternatives.

When the significance level was increased to .12, the relationship between the run types showed a possible trend. As before, the lowest number of errors occured when the satellite used the artificial intelligence technique while the SRT used

the game theory technique (run type (GT,AI)). The highest number of errors occured when the SRT used the artificial intelligence technique (run types (AI,AI) and (AI,GT)); these two run types still reflected coming from the same population. The remaining run type, run type (GT,GT), resulted in a moderate number of errors somewhere in between the other two levels.

A metagame argument could be used to select each player's decision technique. Rearranging the mean values on table 2 results in the metagame matrix shown in figure 14. After removing the dominated options, only the artificial intelligence technique options remain. Figure 15 shows the metagame matrix after accounting for the statistical analysis explained earlier. Again, after removing the dominated strategies, only the artificial intelligence technique options remain. Thus, applying dominance to a metagame argument results in both players selecting the artificial intelligence technique.

| | | SATELLITE AI | SATELLITE GT |
|---|---|---|---|
| SRT | AI | 2.06 | 2.08 |
| | GT | 1.14 | 1.76 |

Figure 14. Metagame Matrix

| | | SATELLITE AI | SATELLITE GT |
|---|---|---|---|
| SRT | AI | 2 | 2 |
| | GT | 1 | 2 |

Figure 15. Revised Metagame Matrix

### 7.5 *Summary*

This chapter discussed an initial comparison of the game theory and artificial intelligence decision techniques. The measure of effectiveness was the number of times the satellite incorrectly predicted the SRT follow-on site. Four run types were used permitting all combinations of the two techniques to be tried. An analysis

of the results showed the artificial intelligence technique produces a better decision when running against the game theory technique. More analysis under varying evaluation mechanisms would give a better understanding of the relationship between the game theory and artificial intelligence techniques. The next chapter discusses the conclusions of this thesis and the recommendations for further study.

# *VIII. Conclusions and Recommendations*

This chapter discusses the conclusions of this thesis and recommendations for further research. The conclusions include discussions of the implementation difficulties of, advantages of, and differences between the two techniques. The recommendations include areas where the comparison can be expanded as well as areas where PALANTIR can be enhanced.

## *8.1 Conclusions*

The purpose of this thesis was to develop a model for the comparison of a game theory decision process to an artificial intelligence decision process. The two-player zero-sum technique was the game theory method used in PALANTIR. The minimax technique with an alpha-beta cutoff heuristic was the artificial intelligence method used. As implemented, the minimax technique was essentially an exact branch and bound algorithm. Minor difficulties were encountered in the implementation of both techniques. These difficulties were overcome and the implementation process completed. In the simulation runs, each technique had advantages over the other. The simulation runs also brought out the differences between the two. These implementation difficulties, advantages, and differences are discussed in the following paragraphs.

*8.1.1 Implementation Difficulties* Minor difficulties were encountered with the implementation of each decision technique. The game theory implementation encountered problems calculating the probabilities of detection for the game matrix and solving this matrix. The artificial intelligence implementation encountered problems in the evaluation of the leaves of the tree and propagating these values to the root node. Of the two, the game theory implementation provided the biggest challenge.

The implementation of the game theory technique had two areas of difficulty: the calculation of the probabilities of detection and solving the game matrix. The probabilities of detection depended on three values: the illumination value, the satellite efficiency value, and the cloud free line of sight (CFLOS) value. The difficult part was obtaining the CFLOS value. A regression equation was developed to obtain the CFLOS for PALANTIR. The equation was based on empirical data. Fitting an equation to this data proved to be a matter of tradeoffs. Appendix A contains more information on the CFLOS equation. Once this equation was developed, the game matrix required a solution method.

The solution of the game matrix is described in Chapter 2. When it involved a pure strategy, the solution implementation presented no problems. When it involved a mixed strategy, the solution implementation required a simplex routine. The coding of this routine proved to be difficult as the game matrices required the use of artificial variables to obtain a basic feasible solution. The artificial intelligence implementation was not as difficult as the game theory implementation.

The artificial intelligence implementation could encounter two areas of difficulty: evaluating the leaves of the tree and propagating these values to the root node. The evaluation of the leaves used the CFLOS equation developed for the game theory technique. Therefore, in this program, the difficulty in implementing the artificial intelligence evaluation method was minimal. However, when the artificial intelligence evaluation method differs from the game theory evaluation method, its implementation is one of the most difficult parts of the artificial intelligence implementation. For this program, the implementation difficulty was in propagating the values of the leaves to the root node.

The propagation of the values of the leaves to the root node was difficult due to the nature of the satellite versus SRT search problem. The evaluation of a specific path and site combination does not depend on the previous moves. The combination will always evaluate to the same value no matter where it is located

in the tree. In the search problem, several of the leaves involve the same path/site combination. Each branch of the tree tends to have the same set of leaves. Thus, when the value of these nodes is propagated up the tree using the straight minimax procedure, the branches take on identical values. All moves appear to be equal in value. Therefore, the minimax procedure was modified to allow all of the nodes along the branch's best route to influence its value. This modification is described in Chapter 5. Determining the implementation of the propagation problem proved to be the most difficult part of the artificial intelligence implementation. The next section describes the advantages of using each decision technique.

*8.1.2 Advantages* Each technique has its advantages over the other. The game theory technique has two major advantages: its ability to provide an exact answer when tradeoffs are involved and its processing speed. The ability to provide exact answers is most useful when two or more alternatives look good for different reasons. The best answer in this case is a distribution over the possible choices. The game theory technique provides this capability through the use of mixed strategy solutions. The other advantage of the game theory technique is its speed. It takes under a second to perform one iteration and under a minute for a simulation run. The artificial intelligence technique, on the other hand, takes approximately five seconds for one iteration and five minutes for one run.

The artificial intelligence technique has three major advantages: its ability to change the number of moves considered for the current move, its ability to use special rules, and its ability to trim the search space. To change the number of moves considered for the current move, the artificial intelligence technique requires only one number to change. The game theory technique, on the other hand, requires a modification to the actual code. The artificial intelligence technique also has the ability to use rules difficult to write in conventional programming languages. An example of such a rule is, "if the site has at least 50 percent cloud coverage and the SRT hasn't moved in the last three decision cycles, then the SRT should move."

The third advantage is the ability to trim the search space. By using the alpha-beta cutoff heuristic, the artificial intelligence technique can cut unproductive moves thereby reducing the search space. In the SRT versus satellite search problem, this reduction ranged from 3 to 50 percent.

*8.1.3 Differences* The main difference between the techniques lies in their handling of situations where two or more alternatives look viable for different reasons (trade-offs exist). When the solutions did not involve a tradeoff situation, the two techniques produced the same results; the satellite correctly predicted the SRT's follow-on site. The differences between the two techniques was only visible when the process involved choosing between alternatives with tradeoffs.

When a tradeoff situation exists, the two techniques differ in their approach. The artificial intelligence technique does not treat the situation any differently than a non-tradeoff situation. It selects the branch with the best value based on its predictions of what the opponent will do. In a sense, it provides an approximate answer to the situation. On the other hand, the game theory technique treats a tradeoff situation differently than a non-tradeoff situation. For a tradeoff situation, it builds a distribution of the alternatives and randomly selects the move based on this distribution (for more detail, refer to mixed strategies in Chapter 2). The game theory technique thereby provides an exact answer to this situation.

Another difference between the two techniques is their representation of the situation. While the game theory technique uses a game matrix to hold the current situation, the artificial intelligence technique uses a game tree. The game matrix approach places one or more moves into one strategy. In other words, the game matrix is an approximate representation of the move decision. The game tree approach enumerates the possible moves in a decision tree; each branch is a different move. There is no lumping together of several moves into one strategy. Thus, the game tree is an exact representation of the current situation.

The game theory technique provided an exact answer to an approximate representation of the situation while the artificial intelligence technique provided an approximate answer to an exact representation of the situation. For the SRT versus reconnaissance satellite search problem, the approximate answer provided slightly better results than the exact answer (one second versus five seconds is irrelevant to a decision cycle of three hours, but would be a factor in analysis simulations). Before the benefits of using the artificial intelligence process over the game theory process can be accurately determined, several areas of comparison require further work.

### *8.2 Recommendations for Further Study*

There are several recommendations for further study. Some involve enhancing the model; some involve enhancing the comparison. Each recommendation is briefly explained below.

1. Allow the SRT and satellite to possess different information. This information could include weather forecasts, satellite fleet information, and unknown SRT operating locations.

2. Develop a different evaluation method for the artificial intelligence game tree. This could involve evaluation of rules that would be hard to write in procedural programming languages such as "if the cloud cover is greater than 50 percent and the SRT hasn't moved for three decision cycles and it is night, then the SRT should move." This would use the artificial intelligence method to its full capability.

3. Add uncertainty to the artificial intelligence technique. The field of uncertainty in artificial intelligence is receiving a lot of attention. Its use within the model would permit an evaluate of its potential to handle various degrees of confidence in the data obtained (such as weather forecasts and satellite efficiency).

4. Add type II errors to the detection process. Currently, only type I errors are accounted for (not detecting the SRT when it is actually at the site). Type II errors could include false detections (detecting the SRT when it is *not* actually there) and multiple detections (detecting the SRT at more than one site).

5. Use classified data in the model. This data could be used to modify and validate the model.

# Appendix A. *The CFLOS Equation*

The Cloud Free Line Of Sight (CFLOS) equation used in the revised PALANTIR involved testing numerous regression equations. This testing resulted in two equations providing close approximations to the empirical data. Selection between the two depended upon the desired tradeoff. One provided values within three percent of the true values but with critical areas possessing the wrong slope. The other provided the proper slope at all values but the estimates only fall within six percent of the true values. Both are based on empirical data from the US Air Force Geophysical Laboratory and the Dynamics Branch at Hanscom AFB, MA.

## *A.1 The Empirical Data*

The Geophysical Laboratory and Dynamics Branch compiled statistics on the cloud cover over various locations in the western US. The data was collected over a ten year period at various times of the year (18:39). Table 3 contains the empirical data produced by their efforts in the form of generic data which can be used for any location in any season.

Table 3. Probability of CFLOS (18:43)

| Sat Elev Angle, $\theta$ (degrees) | Observed Cloud Cover, $\mu$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| 90° | 1.00 | 0.97 | 0.92 | 0.87 | 0.81 | 0.77 | 0.70 | 0.62 | 0.48 | 0.31 | 0.08 |
| 80° | 0.99 | 0.97 | 0.92 | 0.87 | 0.81 | 0.77 | 0.69 | 0.61 | 0.47 | 0.31 | 0.08 |
| 70° | 0.99 | 0.97 | 0.91 | 0.86 | 0.80 | 0.76 | 0.68 | 0.61 | 0.47 | 0.30 | 0.08 |
| 60° | 0.99 | 0.96 | 0.90 | 0.85 | 0.80 | 0.75 | 0.66 | 0.60 | 0.46 | 0.29 | 0.08 |
| 50° | 0.99 | 0.96 | 0.90 | 0.85 | 0.78 | 0.73 | 0.64 | 0.58 | 0.45 | 0.29 | 0.08 |
| 40° | 0.99 | 0.95 | 0.88 | 0.83 | 0.76 | 0.71 | 0.62 | 0.55 | 0.42 | 0.27 | 0.07 |
| 30° | 0.98 | 0.93 | 0.86 | 0.80 | 0.73 | 0.66 | 0.57 | 0.50 | 0.38 | 0.24 | 0.06 |
| 20° | 0.98 | 0.90 | 0.83 | 0.75 | 0.67 | 0.59 | 0.50 | 0.42 | 0.33 | 0.21 | 0.05 |
| 10° | 0.97 | 0.86 | 0.76 | 0.65 | 0.55 | 0.47 | 0.39 | 0.32 | 0.24 | 0.16 | 0.03 |

### *A.2 The Original CFLOS Equation*

The first CFLOS equation evaluated was the one used in the original PALANTIR. Regression analysis indicated it was one of two which provided the closest match to the empirical data. The equation is:

$$CFLOS(\theta, \mu) = 0.77653 + 0.00652\theta - 0.00004\theta^2 - 0.22584\mu - 0.62821\mu^2 \qquad (3)$$

where

$\theta$ = Elevation angle of the satellite, and

$\mu$ = Percentage of cloud cover over the site.

Tests were performed comparing the output of this equation to the empirical data in Table 3. Eq (3) produced values which varied less than three percent from the empirical data. The problem with this equation comes from the slope of the line for the range of elevation angles critical to the program.

When determining the CFLOS value for follow-on sites, over 70 percent of the elevation angles fell between 75° and 90°. For these elevation angles, the empirical data shows either no slope or a slight positive slope. With Eq (3), over 45 percent of the tested values showed a negative slope between 80° and 90°. When this negative slope occurs for the site being evaluated, the highest CFLOS value is at the follow-on site furthest away. This results in the highest probability of detection being at the site furthest away, contrary to the real world.

### *A.3 The Revised CFLOS Equation*

The alternate CFLOS equation evaluated was discovered using regression analysis. Regression analysis indicated it was one of two which provided the closest match to the empirical data. The equation is:

$$CFLOS(\theta,\mu) = 0.88964 + 0.0021217\theta - 0.75599\mu + 0.77157\mu^2 - 0.93593\mu^3 \quad (1)$$

where

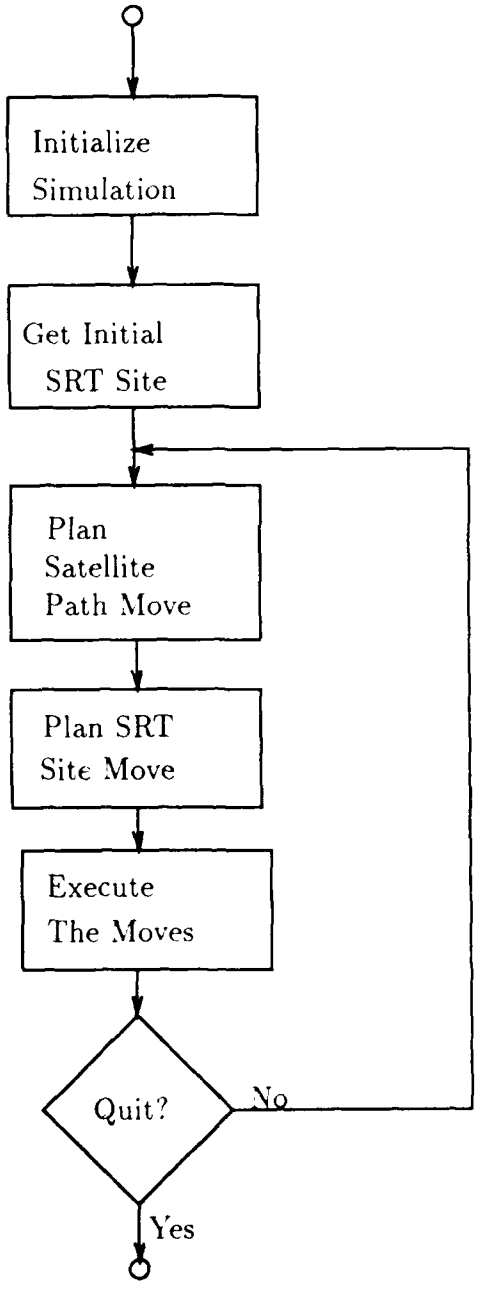
$\theta$ = Elevation angle of the satellite, and

$\mu$ = Percentage of cloud cover over the site.

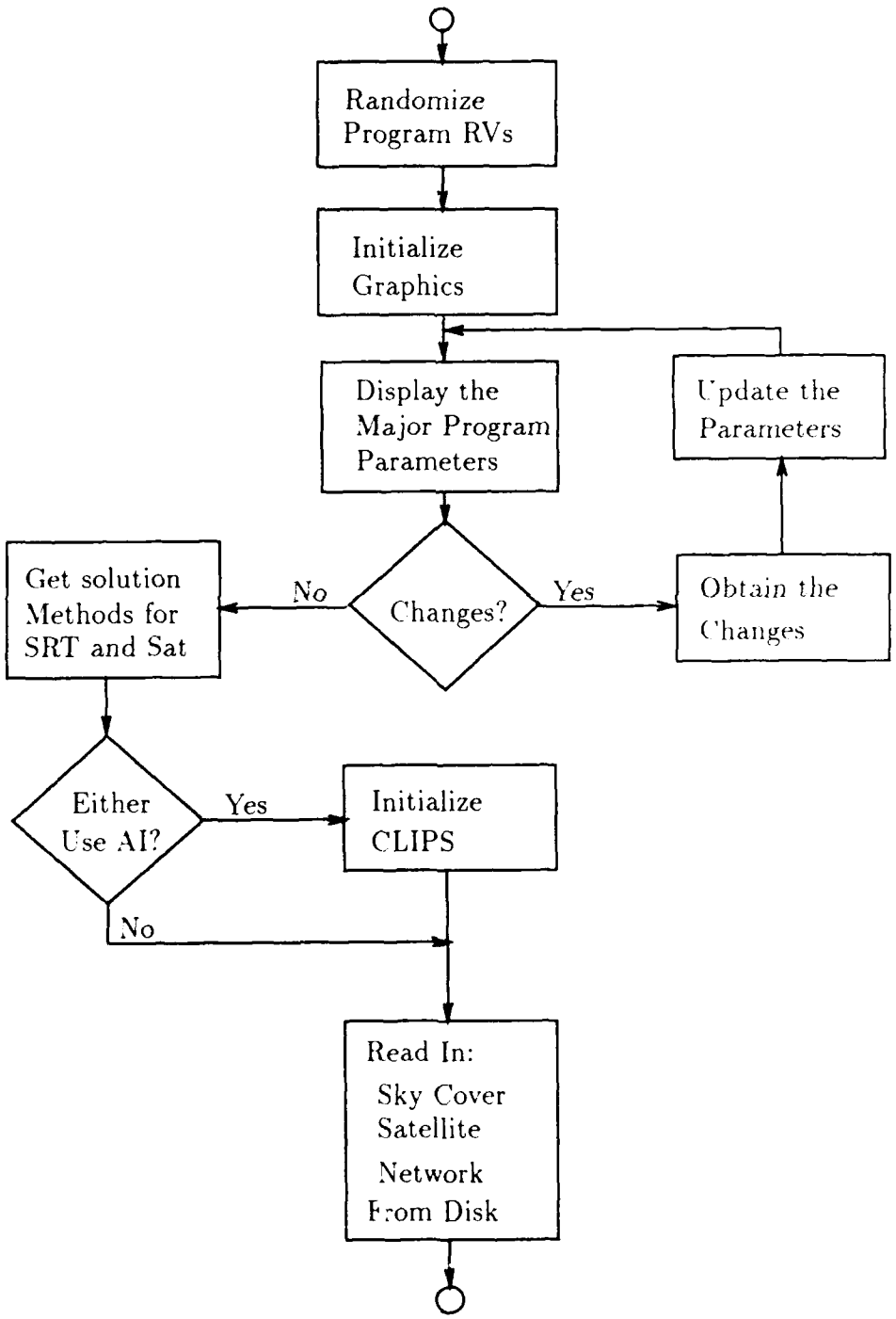
Tests were performed comparing the output of Eq (1) to the empirical data in Table 3. The equation produced values with a slope closely related to the slope of the empirical data. The slope of the equation values is either zero or positive. Thus, the highest CFLOS value (and therefore the highest probability of detection) is at the closest site. This matches the real world system. The problem with this equation is the values varied from the empirical data up to six percent, twice as much as Eq (3). This problem is greatly reduced because over 60 percent of the time, the two slopes intersect in the 70° to 85° region. When the intersection occurs in this region, the equation's CFLOS value is far closer to the empirical value than six percent, rivalling the two percent of Eq (3). Because of the closely matching slope and the high percentage of intersections in the 70° to 85° region, Eq (1) was incorporated into the revised PALANTIR.

# Appendix B. *Appendix B. Top Level Flowcharts*

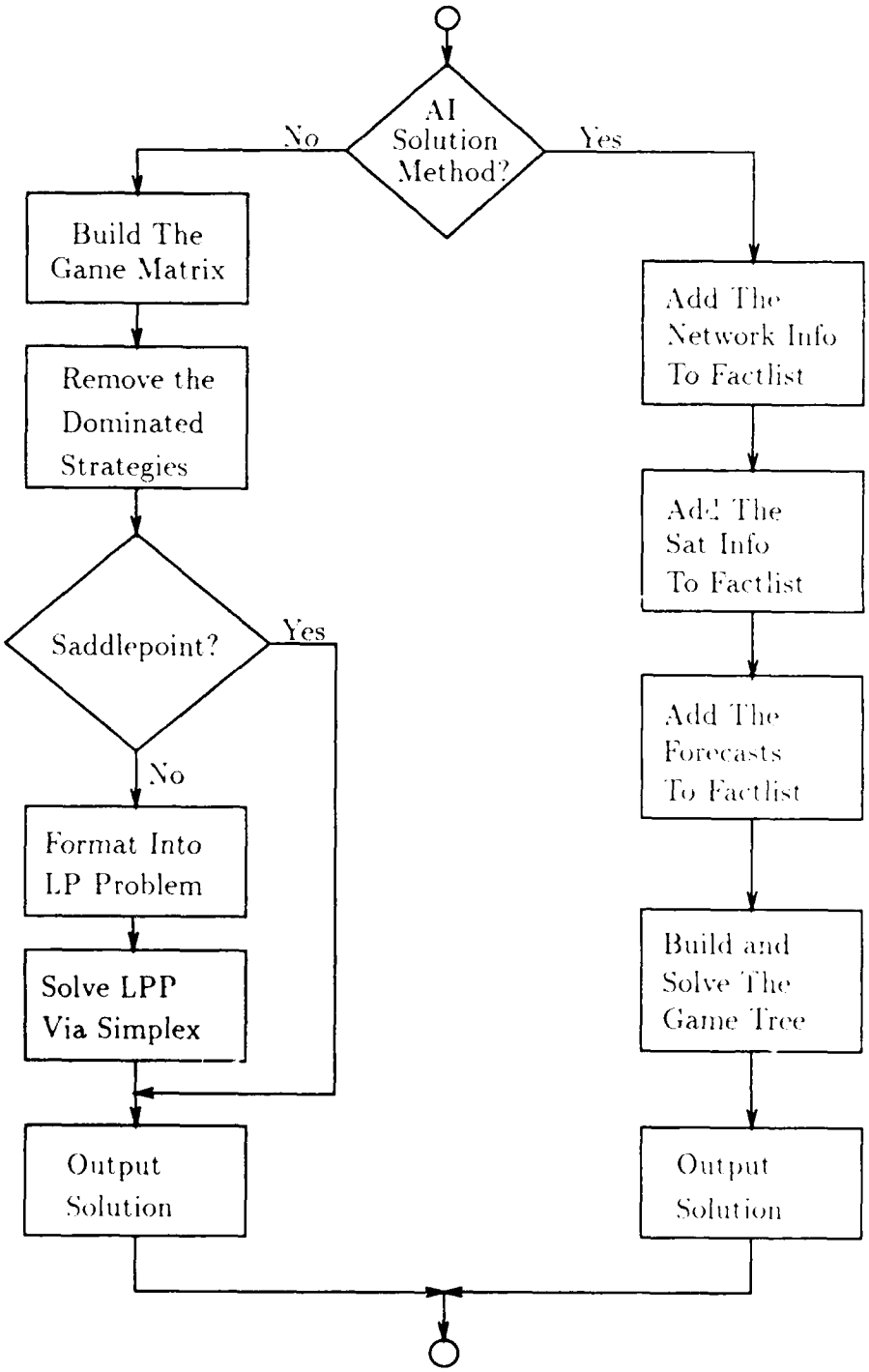
## *B.1 Overview FlowChart*



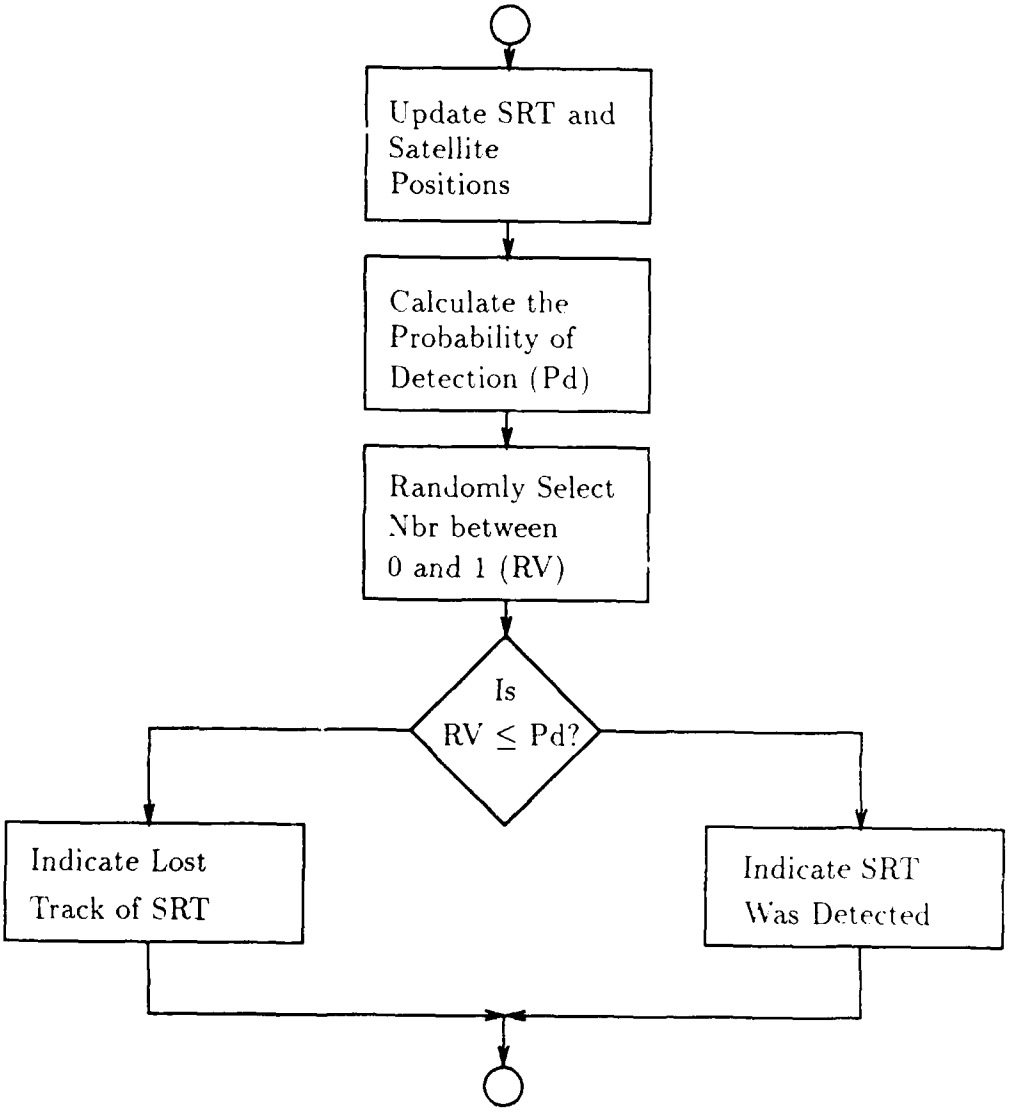
## *B.2 Initialization Flowchart*



## *B.3 Sateilite/SRT Planning Flowchart*



*B.4 Move Execution Flowchart*



# Appendix C. *Initial Facts for an AI Solution*

This appendix describes the initial facts required to build and solve an artificial intelligence game tree. There are four sets of facts required in the fact list prior to running the CLIPS routines: setup facts, network facts, satellite facts, and weather facts. The general format of each is listed below.

## *C.1 Setup Facts*

(Last Known Site ___ Age ___)

(Current Time ___)

(Time Increment ___)

(Min Illumination ___)

(Max Paths ___)

(Max Ply ___)

## *C.2 Network Facts*

(Site ___ Coords ___ ___)

(Site ___ Branches ___ ___ . . . ___)

## *C.3 Satellite Facts*

(Sat Fleet Alt ___ MinProbDetect ___ Efficiency ___)

(Sat ___ Day Slope ___ OverFlightTime ___)

(Sat ___ Night Slope ___ OverFlightTime ___)

(Sat ___ Day Path Intercepts ___ ___ ___ ___ ___)

(Sat ___ Night Path Intercepts ___ ___ ___ ___ ___)

## *C.4 Weather Fact*

(Site ___ Forecast ___ ___ ___ ___ ___ ___ ___ ___ ___ ___)

# *Bibliography*


1. Air University. *Space Handbook.* AU-18. Maxwell AFB, AL: Air University Press, January 1985.

2. Battilega, John A. and Judith K. Grange. *The Military Applications of Modelling.* Wright-Patterson AFB, OH: Air Force Institute of Technology Press, 1984.

3. Bratko, Ivan. *Prolog Programming for Artificial Intelligence.* Reading MA: Addison-Wesley Publishing Company, 1986.

4. Firebaugh, Morris W. *Artificial Intelligence, A knowledge-Based Approach.* Boston: Boyd & Fraser Publishing Company, 1988.

5. Heier, Capt Jeffrey E. *ADAM, Acquisition Deployment and Maneuvering, The Space Game.* MS thesis, AFIT/GSO/ENS/87D-7. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1987 (AD-A189 512).

6. Hillier, Frederick S. and Gerald J. Lieberman. *Introduction to Operations Research* (Fourth Edition). Oakland CA: Holden-Day Incorporated, 1988.

7. Lehner, Paul E. and Others. *Combining decision Analysis and Artificial Intelligence Techniques: An Intelligent Aid for Estimating Enemy Courses of Action,* January 1984 - March 1985. Contract MDA903-83-C-0311. McLean VA: PAR Technical Corporation, August 1985 (AD-A159 846).

8. Lukasik, S. J. "Holding Strategic Targets at Risk." Presented at the AIAA 3rd National C$^3$I Policy Conference. United States Naval Postgraduate School, Monterey, CA, 3 February 1987.

9. Lutz, Major Rollin J. *Application of Artificial Intelligence in a Conflict Analysis Decision Aid.* MS thesis, AFIT/ GST/ENS/87J-10. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, June 1987 (AD-A185 493).

10. Myers, Raymond H. and Ronald E. Walpole. *Probability and Statistics for Engineers and Scientists* (Second Edition). New York: McMillan Publishing Company, 1978.

11. Palmer, Capt Robert J. *A Methodology Employing Competitive Strategies for Predicting Possible Strategic Relocatable Target Movements.* MS thesis, AFIT/GST/ENS/89M-16. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1989 (AD-B131 142).

12. Pearl, J. "Game Trees," *Encyclopedia of Artificial Intelligence*, Volume 1, edited by Stuart C. Shapiro. New York: John Wiley & Sons, 1987.

13. Pritsker, A. Alan B. *Introduction to Simulation and SLAM II* (Third Edition). New York: Halsted Press, 1986.

14. Richmond, Samual B. *Operations Research for Management Decisions*. New York: The Ronald Press Company, 1968.

15. Rivest, Ronald L. "Game Tree Searching by Min/Max Approximation," *Artificial Intelligence, Vol 34*: 77-96 (December 1987).

16. Schwartz, Richard I. and Robert E. Shostak. "The AI Challenge," *PC Tech Journal, Volume 4, No. 9*: 195-197 (September 1986).

17. Thurman, LTG William E. "Challenge of the Future: Harnessing Artificial Intelligence," *Signal 41*: 32-36 (April 1987).

18. Toor, J.S. and G.S. Hans. *Effects of Cloud Cover on Surveillance of Mobile Small ICBM*. Report BMO TR-86-12. Contract F04704-85-C-0154. Encinitas, CA: Dynamic Analusis & Testing Associates (DATA), February 1986 (AD-B104 825).

19. Vane, Russel and others. "Game Playing and Game Trees in Knowledge-Based Adversial Planning," *Sixth Annual Workshop on Command and Control Decision Aiding*: G-1 through G-5. San Diego CA, 1989.

## *Vita*

Captain Paul R. Andreé [redacted] He graduated from Siuslaw High School in Florence, Oregon, in 1978 and joined the US Air Force in October 1979. After basic training, he worked as a computer operator for the 27$^{th}$ TFW at Cannon AFB, Clovis, New Mexico. In 1981, he was accepted into the Airmen's Educational Commissioning Program and attended the University of South Florida in Tampa, Florida, where he received the degree of Bachelor of Science in Computer Science in December 1984. Upon graduating, he attended the Officer Training School at Lackland AFB, San Antonio, Texas, and was commissioned on 5 April 1985. He served as a communication officer for the 1916$^{th}$ Communications Squadron at Pease AFB, New Hampshire, until entering the School of Engineering, Air Force Institute of Technology, in June 1988.

Permanent address: 529 East Wescott Drive
Phoenix, Arizona 85324

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release;<br>distribution unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>AFIT/GSO/ENS/89D-1 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>School of Engineering<br>6b. OFFICE SYMBOL (*If applicable*)<br>AFIT/ENS | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (*City, State, and ZIP Code*)<br>Air Force Institute of Technology<br>Wright-Patterson AFB OH 45433-6583 | 7b. ADDRESS (*City, State, and ZIP Code*) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>8b. OFFICE SYMBOL (*If applicable*) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (*City, State, and ZIP Code*) | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO |

11. TITLE (*Include Security Classification*)
A Model For Comparing Game Theory And Artificial Intelligence Decision Making Processes

12. PERSONAL AUTHOR(S)
Paul R. André, B.S., Captain, USAF

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (*Year, Month, Day*) | 15. PAGE COUNT |
|---|---|---|---|
| MS Thesis | FROM _____ TO _____ | 1989 December | 70 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Game Theory; Computerized Simulation<br>Artificial Intelligence; Minimax Technique |
| 12 | 04 | | |
| 12 | 09 | | |

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)
Thesis Advisor: Bruce W. Morlan, Major, USAF
Instructor of Operational Research
Department of Operational Sciences

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Bruce W. Morlan, Instructor | 22b. TELEPHONE (*Include Area Code*)<br>(513)255-3362<br>22c. OFFICE SYMBOL<br>ENS |

DD Form 1473, JUN 86 *Previous editions are obsolete*

*Abstract*

The purpose of this study was to develop an analytical tool to compare the use of an artificial intelligence decision making method to a game theory decision making method. The game theory method used was the two-player zero-sum gaming technique. The artificial intelligence method used was the minimax technique with an alpha-beta cutoff heuristic. The simulation program was written in the C programming language. The artificial intelligence method made calls to a forward chaining shell called C Language Production System, CLIPS.

The original game theory method was developed by Capt Robert Palmer. It was written in Turbo Pascal and required some modification before use in this study. The program was converted to the C language, modified to fit its intended use, and fit with the artificial intelligence method described above.

The program played reconnaissance satellites against ground mobile Strategic Relocatable Targets (SRTs). It implemented the methods described within this document to allow for a comparison of the two techniques. For an initial comparison, the measure of effectiveness was the number of times the satellite incorrectly predicted the SRT's movement.